AD-A118 414 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 17/5
A TARGET ACQUISITION MODULE FOR THE STAR COMBINED ARMS COMBAT S--ETC(U)
APR 82 J K HARTMAN MIPR-CD-2-82
UNCLASSIFIED NPS55-82-014 NL

1 OF 2
AD A
118414

②

AD A118414

NPS55-82-014

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A TARGET ACQUISITION MODULE
FOR THE
STAR COMBINED ARMS COMBAT SIMULATION
VOLUME II
TECHNICAL MANUAL

by

James K. Hartman

April 1982

Approved for public release; distribution unlimited.

Prepared for:
US Army Training and Doctrine Command, Fort Monroe, VA 23651

82 08 20 046

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

Rear Admiral J. J. Ekelund
Superintendent

D.A. Schrady
Acting Provost

This work was supported with funds provided by the US Army Training and Doctrine Command, Fort Monroe, Virginia 23651

James K Hartman

James K. Hartman
Department of Operations Research

Reviewed by:

Kneale T. Marshall

Kneale T. Marshall, Chairman
Department of Operations Research

William M. Tolles

William M. Tolles
Dean of Research

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS55-82-014	2. GOVT ACCESSION NO. AD-A114 424	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Target Acquisition Module for the STAR Combined Arms Combat Simulation, Volume II, Technical Manual		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James K. Hartman		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS MIPR# CD 2-82
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Training and Doctrine Command Fort Monroe, Virginia 23651		12. REPORT DATE April 1982 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) U 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Target Acquisition, Detection, Search, STAR, Combat Models, Simulation, NVL, Electrooptics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report provides technical documentation for the Target Acquisition Module of the STAR Combined Arms Combat Simulation Model. Details of data structures, simulation events, and supporting routines are provided.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

I. INTRODUCTION

The STAR Target Acquisiton Module has been developed to enable the STAR (Simulation of Tactical Alternative Responses) combat simulation model to simulate various ways of using modern sensor devices for battlefield target acquisition. Major features of the target acquisition module are:

1. A wide variety of electro-optical and thermal imaging sensor devices can be modelled using the Night Vision Laboratory (NVL) methodology. (DYNTACS/ASARS visual detection models are also available.)

2. Each combat vehicle may have several observers.

3. Each observer may use multiple sensors.

4. There is no model-imposed limit on the number of different vehicle - observer - sensor assignments.

5. Several flexible "search tactics" are defined to model the detailed employment of the sensor devices by each observer.

6. The model includes degradation of target acquisition due to smoke, weather, and nightime conditions.

7. Various levels of acquisition are modelled, with the potential for investigating effects of limited information on target selection and weapon employment tactics.

A summary of the target acquisition module is given in Volume I of this report (Reference 1). It is assumed that the reader has completed Volume I before attempting to read this report. Most of the descriptive matter of the summary volume will not be repeated here.

The current document (Volume II) concentrates on details of data structures, events, and routines which implement the target acquisition module in SIMSCRIPT II.5 code. Chapter II presents the STAR implementation of the U.S. Army Night Vision and Electro-Optical Laboratory (NVL) sensor device models for target acquisition. Chapter III details the modified STAR target list structure which incorporates acquisition level information for each target. Chapter IV presents the data structures which are used to associate sensor devices and search tactics with individual simulated observers. In Chapter V we analyze the SEARCH event and its interaction with other STAR combat functions. Chapter VI presents details of the search tactics routines currently implemented in STAR.

II. STAR IMPLEMENTATION OF THE NVL DETECTION MODEL

A. SCOPE OF THE NVL MODEL - INPUT PARAMETERS AND OUTPUT

Chapter III of Volume I of this report (Reference 1) presents a summary of the NVL target detection methodology. The NVL model assumes that we have identified an observer and a potential target, that we have selected a sensor device and the mode in which it will be used, that we have described the observer's field of search, and that we know the level of acquisition which the observer is attempting to attain. Given these input parameters, the NVL mdoel computes (as described in Volume I, Chapter III) whether an acquisition is possible, and, if so, a stochastic time to acquire the target.

Determination of the above listed input parameters is outside the scope of the NVL methodology and must be handled by the rest of the STAR target acquisition module (in particular the search tactics routines). Similarly the use of the resulting time to acquire is extraneous to the NVL model. In this chapter we will consider only the NVL Model, leaving for later chapters its interaction with the rest of the target acquisition module.

B. ROUTINE NVL.DET

All NVL acquisition time computations in STAR are performed by a call to the routine NVL.DET with input parameters:

A	-	Pointer to observer
B	-	Pointer to potential target
SENSOR	-	Sensor to be used
MODE	-	Code for Sensor Mode of use, (Typically wide or narrow field of view)

LO.ACQ.LEV - Lowest acceptable acquisition level

HI.ACQ.LEV - Highest acquisition level to try for

HFOS - Observer horizontal field of search

VFOS - Observer vertical field of search

MAXTIME - Max time to spend trying to acquire target

and yielding:

TIME - Time to acquire target (or RINF.C if acquisition does not occur)

ACQ.LEV - Acquisition level achieved

The SIMSCRIPT code for routine NVL.DET is given in Figures II-1. The following comments in conjunction with the description in Volume I, Chapter III should make its operation clear. NVL.DET is a driver routine which calls a number of supporting routines. These supporting routines will be documented in Section C of this Chapter. Data arrays used will be discussed in Section D.

There are two possible exits from routine NVL.DET. The normal exit at Line 23 (See Figure II1) returns a computed time-to-acquire, while the "QUIT" exit (Lines 76-80) is used whenever acquisition is impossible and returns TIME=RINF.C and ACQ.LEV=0.

The routine has two phases. In the first, optimistic assumptions are made to avoid LOS, background, and smoke computations. If target acquisition under these conditions does not occur, then the routine returns. Otherwise, the second phase computes LOS, target background, and smoke attenuation for a more accurate target acquisition assessment.

GIVEN ARGUMENTS

A	INTEGER	Pointer to observer entity
B	INTEGER	Pointer to target entity
SENSOR	INTEGER	Sensor Code
MODE	INTEGER	Sensor Mode of use code
LO.ACQ.LEV	INTEGER	Lowest acceptable acquisition level
HI.ACQ.LEV	INTEGER	Highest acquisition level to try for
HFOS	REAL	Angle of observer's horizontal field of search
VFOS	REAL	Angle of observer's vertical field of search
MAXTIME	REAL	MAX time to spend trying to acquire target

YIELDING ARGUMENTS

TIME	REAL	Computed time to acquire target
ACQ.LEV	INTEGER	Acquisition level achieved (or zero if no acquisition occurs)

LOCAL VARIABLES

DEV	INTEGER	NVL device code
RANGE	REAL	Observer to target range
PCT.VIS	REAL	Fraction of target vertical height visible to observer
BACKGRND	INTEGER	Target background code
SPECTRUM	INTEGER	Wavelength band code
SIGNATURE	REAL	Target Signature
SM.ATTEN	REAL	Attenuation factor due to smoke clouds

SENSR.INPUT	REAL	Attenuated target signature
SPATIAL.FREQ	REAL	Spatial frequency, F
CYCS	REAL	Resolution cycles on target image
N50	REAL	NVL Johnson Criterion n50
CYCLE.RATIO	REAL	N/n50 Ratio
PROB.INF	REAL	P_∞
DISTOBKG	REAL	Rough estimate of distance to background
RN	REAL	Uniform (0,1) Random Number
RATE	REAL	Search Rate $1/\tau_1$

GLOBAL VARIABLES (See Section D for Further Explanation)

SENSR.PARS	REAL 3-D	Sensor device parameters
TARDIM	REAL 3-D	Combat entity dimensions
RINF.C	DOUBLE	System defined infinity
MX.SIG	REAL 3-D	Largest that tgt signature can be
CRITICAL.VALUE	REAL	Threshold on PCT.VIS below which no acquisition can occur
N.SMK.SET	INTEGER	Number of smoke clouds currently on battlefield

ENTITY ATTRIBUTES

SYS.TYPE	INTEGER	System type of target
WPN.TYPE	INTEGER	Weapon type of target

ROUTINES AND FUNCTIONS CALLED

DIST

ATTENUATION

UNIFORM.F

RESOLUTION

JOHNSON.CRITERION

PR.INFINITY

LOS.GEOM

SMK.ATTEN

TGT.SIGNATURE

SCH.RATE

LOG.E.F

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

SEE FIGURE II-I

LINE BY LINE COMMENTARY (NVL.DET)

Lines 1 - 19	Give the routine declaration and variable definitions.
Lines 20 - 23	Screen out targets which are outside the sensor device's maximum range.
Lines 24 - 27	Make optimistic assumptions about target signature and attenuation (to avoid LOS, Background, and Smoke computations).
Lines 28 - 43	Compute acquisition under the optimistic Phase I assumptions. For more detail see the discussion of Phase 2 below. If acquisition is possible at any of the allowed levels, we go to Phase 2, otherwise QUIT.
Lines 44 - 47	Begin the Phase 2 computations.
Lines 48 - 51	Call Routine LOS.GEOM to determine whether line of sight to the target exists. If so, the percent of the target's vertical height which is visible is computed along with the target background code.

Lines 52 - 54	Call SMK.ATTEN to compute the target signature attenuation factor due to smoke clouds between observer and target.
Line 55	Calls routine TGT.SIGNATURE to get the target signature.
Lines 56 - 59	Call routine ATTENUATION to attenuate the target signature to account for atmospheric effects and smoke between the observer and the target. If the resulting input to the sensor is less than the sensor minimum sensitivity threshold go to "QUIT".
Line 60	Calls routine RESOLUTION to compute the MRT (MRC) function for the sensor.
Lines 63 - 75	Loop through each allowable acquisition level. Acquisition is tested for each level, the highest possible level being returned at Line 73. For each level, the following computations are performed:
Lines 64 - 65	Call Routine JOHNSON.CRITERION to look up the cycle threshold n50 for this detection situation and then compute the cycle ratio N/n50.
Line 66	Calls Routine PR.INFINITY to compute the P_∞ value using the target transform probability function.
Lines 70 - 73	Call Routine SCH.RATE to compute the search rate and use it along with RN to compute the final acquisition time for return to the calling program.
Lines 76 - 79	Are branched to whenever acquisition becomes impossible, returning an infinite time to the calling program.

```
ROUTINE NVL.DET
''*********************************
GIVEN       A,           ''OBSERVER POINTER
            B,           ''TARGET POINTER
            SENSOR,      ''NVL DEVICE NUMBER
            MODE,        ''MODE OF USE
            LO.ACQ.LEV,  ''LOWEST ACCEPTABLE ACQUISITION LEVEL
            HI.ACQ.LEV,  ''HIGHEST ACQUISITION LEVEL TO TRY FOR
            HFOS,        ''OBSERVER HORIZONTAL FIELD OF SEARCH
            VFOS,        ''OBSERVER VERTICAL FIELD OF SEARCH
            MAXTIME      ''MAX TIME TO SPEND TRYING TO ACQUIRE
YIELDING    TIME,        ''TIME TO ACQUIRE TARGET
            ACQ.LEV      ''ACQUISITION LEVEL ACHIEVED
''  NVL METHODOLOGY FOR TARGET ACQUISITION TIMES
DEFINE A, B, SENSOR, ACQ.LEV, BACKGRND, MODE, ANSWER, SPECTRUM,
    LO.ACQ.LEV, HI.ACQ.LEV, DEV  AS INTEGER VARIABLES
DEFINE TIME, RANGE, PCT.VIS, SIGNATURE, SENSR.INPUT, SM.ATTEN,
    SPATIAL.FREQ, N50, CYCLE.RATIO, PROB.INF, RATE, RN, HFOS, VFOS,
    DISTOBKG, CYCS, MAXTIME AS REAL VARIABLES
LET RANGE = DIST(A,B)
IF RANGE GT SNSR.PARS(SENSOR,MODE,1)    ''MAX RANGE FOR SENSOR
    GO TO 'QUIT'
OTHERWISE   ''FIRST DO OPTIMISTIC COMPUTATION WITHOUT LOS AND SMOKE
LET SPECTRUM = SNSR.PARS(SENSOR,MODE,3)
LET DEV = SNSR.PARS(SENSOR,MODE,9)   ''NVL DEVICE CLASS
LET SIGNATURE = MX.SIG(SYS.TYPE(B),WPN.TYPE(B),SPECTRUM)
LET SM.ATTEN = 0.0
CALL ATTENUATION(SIGNATURE,SPECTRUM,RANGE,SM.ATTEN) YIELDING SENSR.INPUT
IF SENSR.INPUT LT SNSR.PARS(SENSOR,MODE,2)    ''MINIMUM SENSOR INPUT
    GO TO 'QUIT'
OTHERWISE
LET RN = UNIFORM.F(0.0,1.0,7)
CALL RESOLUTION(SENSOR,MODE,SENSR.INPUT) YIELDING SPATIAL.FREQ
LET CYCS = 1000.0*SPATIAL.FREQ*TARDIM(SYS.TYPE(B),WPN.TYPE(B),4)/RANGE
FOR ACQ.LEV BACK FROM HI.ACQ.LEV TO LO.ACQ.LEV DO
    CALL JOHNSON.CRITERION(DEV,ACQ.LEV,B) YIELDING N50
    LET CYCLE.RATIO = CYCS / N50    ''ASSUMING PCT.VIS = 1.0
    CALL PR.INFINITY(CYCS,N50) YIELDING PROB.INF
    IF RN LT PROB.INF
        GO TO 'LOS.CHECKS'
    OTHERWISE
LOOP
GO TO 'QUIT'    ''SINCE NO ACQ.LEV IS POSSIBLE
'LOS.CHECKS'
''NOW DO LOS AND SMOKE COMPUTATIONS
LET HI.ACQ.LEV = ACQ.LEV
```

FIGURE II-1   ROUTINE NVL.DET

```
LET DISTOBKG = SNSR.PARS(SENSOR,MODE,8)   ''DIST PAST TGT TO CHECK FOR BKG
CALL LOS.GEOM(A,B,SPECTRUM,DISTOBKG) YIELDING PCT.VIS, BACKGRND, DISTOBKG
IF PCT.VIS LE CRITICAL.VALUE
    GO TO 'QUIT'
OTHERWISE
IF N.SMK.SET GT 0
    CALL SMK.ATTEN(A,B,SPECTRUM,PCT.VIS) YIELDING SM.ATTEN
ALWAYS
CALL TGT.SIGNATURE(B,BACKGRND,SPECTRUM) YIELDING SIGNATURE
CALL ATTENUATION(SIGNATURE,SPECTRUM,RANGE,SM.ATTEN) YIELDING SENSR.INPUT
IF SENSR.INPUT LT SNSR.PARS(SENSOR,MODE,2)    ''MINIMUM SENSOR INPUT
    GO TO 'QUIT'
OTHERWISE
CALL RESOLUTION(SENSOR,MODE,SENSR.INPUT) YIELDING SPATIAL.FREQ
LET CYCS = PCT.VIS*1000.0*TARDIM(SYS.TYPE(B),WPN.TYPE(B),4)*
    SPATIAL.FREQ/RANGE
FOR ACQ.LEV BACK FROM HI.ACQ.LEV TO LO.ACQ.LEV DO
    CALL JOHNSON.CRITERION(DEV,ACQ.LEV,B) YIELDING N50
    LET CYCLE.RATIO = CYCS / N50
    CALL PR.INFINITY(CYCS,N50) YIELDING PROB.INF
    IF RN GE PROB.INF
        CYCLE
    OTHERWISE
    CALL SCH.RATE(SENSOR,MODE,CYCLE.RATIO,HFOS,VFOS) YIELDING RATE
    LET TIME = LOG.E.F(1.0-RN/PROB.INF)/(-RATE)
    IF TIME LE MAXTIME
        RETURN   ''WITH TIME AND ACQ.LEV
    OTHERWISE
LOOP
'QUIT'   ''ACQUISITION WILL NOT OCCUR
LET TIME = RINF.C
LET ACQ.LEV = 0
RETURN
END
```

FIGURE II-1   (CONTINUED)

C. SUPPORTING ROUTINES

This section documents routines which are called by NVL.DET to perform parts of the NVL methodology.

1. Routine LOS.GEOM. Routine LOS.GEOM is responsible for two functions. First it computes geometric line of sight between the observer and the target taking into account the STAR terrain and forest features. If geometric line of sight exists, it then computes the target background as seen by the observer.

GIVEN ARGUMENTS

A	INTEGER	Pointer to observer entity
B	INTEGER	Pointer to target entity
SPECTRUM	INTEGER	Sensor wavelength band code
MAXDIST	REAL	Distance beyond target to check for background.

YIELDING ARGUMENTS

PCT.VIS	REAL	Fraction of target height visible to observer.
BACKGRND	INTEGER	Target background type code
DISTOBKG	REAL	Rough estimate of distance from target to background

LOCAL VARIABLES

NONE

GLOBAL VARIABLES

FWD.LOOK	INTEGER	Inputs to control LOS routine
BWD.LOOK	INTEGER	

VISFRB.LS	REAL	Percent Visible output from LOS routine
MXBKGND	INTEGER	Number of background codes allowed for in target signature data base
N.SMK.SET	INTEGER	Number of smoke clouds on the battlefield
CRITICAL.VALUE	REAL	Threshold for PCT.VIS below which acquisitions are not allowed.

ENTITY ATTRIBUTES

NONE

ROUTINES AND FUNCTIONS CALLED

SIGHT

LOS.BKGND

SMK.NEAR.BKGND

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-2

LINE BY LINE COMMENTARY (LOS.GEOM)

Lines 1 - 5	Give routine DECLARATION and VARIABLE definitions.
Lines 6 - 9	Call STAR Routine SIGHT to do the geometric line of sight computations.
Lines 11 - 17	Compute the target background code as seen by the observer. If only one background is allowed for in the target signature data base, then the computations are skipped.

Note that routine SIGHT is a standard STAR routine, LOS.BKGND is documented in Section E of this Chapter, and routine SMK.NEAR.BKGND is documented in the STAR Smoke Model Report (Reference 2).

See Section E of this Chapter for the definition of the background codes used by the STAR model.

```
ROUTINE LOS.GEOM(A,B,SPECTRUM,MAXDIST) YIELDING PCT.VIS,BACKGRND,DISTOBKG
''*****************************************************
'' COMPUTE LOS, AND IF LOS EXISTS, COMPUTE TARGET BACKGROUND FOR NVL.DET
DEFINE A, B, SPECTRUM, BACKGRND AS INTEGER VARIABLES
DEFINE PCT.VIS, MAXDIST, DISTOBKG AS REAL VARIABLES
LET FWD.LOOK = 1
LET BWD.LOOK = 0
CALL SIGHT(A,B)
LET PCT.VIS = VISPRB.LS
LET BACKGRND = 1
IF PCT.VIS GE CRITICAL.VALUE AND MXBKGND GE 2
     CALL LOS.BKGND(MAXDIST) YIELDING BACKGRND, DISTOBKG
     IF N.SMK.SET GT 0  AND (MXBKGND EQ 3 OR MXBKGND EQ 5)
          CALL SMK.NEAR.BKGND(A,B,SPECTRUM,PCT.VIS,BACKGRND,DISTOBKG)
               YIELDING BACKGRND
     ALWAYS
ALWAYS
RETURN
END
```

FIGURE II-2   ROUTINE LOS.GEOM

2. Routine TGT.SIGNATURE. The TGT.SIGNATURE routine looks up optical contrast or temperature difference depending on the sensor used. The current routine is a simple table lookup.

GIVEN ARUGMENTS

B	INTEGER	Pointer to target entity
BACKGRND	INTEGER	Target background type code
SPECTRUM	INTEGER	Sensor wavelength band code

YIELDING ARGUMENT

SIGNATURE	REAL	The target signature

LOCAL VARIABLES

SYS	INTEGER	System type of target
WPN	INTEGER	Weapon type of target

GLOBAL VARIABLES

TAR.SIG	REAL 4-D	Target signature data array

ENTITY ATTRIBUTES

SYS.TYPE	INTEGER	System type of target
WPN.TYPE	INTEGER	Weapon type of target

ROUTINES AND FUNCTIONS CALLED

NONE

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-3

COMMENTARY (TGT.SIGNATURE)

The Routine is Self-Explanatory.

```
ROUTINE TGT.SIGNATURE(B,BACKGRND,SPECTRUM) YIELDING SIGNATURE
''*************************************************************
''GIVEN TARGET B YIELDS CONTRAST FOR OPTICAL DEVICES AND TEMPERATURE
''   DIFFERENCE FOR THERMAL DEVICES -- FOR NVL DETECTION MODEL
DEFINE B, BACKGRND, SPECTRUM,SYS, WPN AS INTEGER VARIABLES
DEFINE SIGNATURE AS A REAL VARIABLE
LET SYS = SYS.TYPE(B)
LET WPN = WPN.TYPE(B)
LET SIGNATURE = TAR.SIG(SYS,WPN,SPECTRUM,BACKGRND)
RETURN
END
```

FIGURE II-3   ROUTINE TGT.SIGNATURE

3. <u>Routine ATTENUATION</u>. The ATTENUATION routine modifies the target signature to account for transmission through the atmosphere and through battlefield smoke clouds along the path from target to sensor.

GIVEN ARGUMENTS

SIGNATURE	REAL	Target signature prior to attenuation
SPECTRUM	INTEGER	Sensor wavelength band code
RANGE	REAL	Observer/Target distance
SMK.ATTEN	REAL	Attenuation factor due to smoke clouds

YIELDING ARGUMENT

SENSR.INPUT	REAL	Attenuated target signature at sensor

LOCAL VARIABLES

SIG.ATMOS	REAL	Attenuation coefficient in open air
ATTEN.FACT	REAL	Atenuation factor due to smoke and atmosphere

GLOBAL VARIABLES

ATMOS.ATTEN	REAL 1-D	Attenuation coefficients
SKY.GROUND	REAL	Sky/Ground brightness ratio for optical systems

ENTITY ATTRIBUTES

NONE

ROUTINES AND FUNCTIONS CALLED

EXP.F

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-4.

COMMENTARY (ATTENUATION)

This routine is largely self-explanatory.

Lines 17 - 20 Prevent numeric overflow in extreme cases where the attenuation factor is very large.

Line 21 Distinguishes between optical devices (SPECTRUM = 1) and thermal devices (SPECTRUM greater than 1).

Lines 22 - 24 Implement the appropriate equation.

```
ROUTINE ATTENUATION
''**************************************************************
GIVEN     SIGNATURE,        ''TARGET SIGNATURE AT TARGET LOCATION
          SPECTRUM,         ''SENSOR WAVELENGTH CODE
          RANGE,            ''OBSERVER TO TARGET RANGE
          SMK.ATTEN         ''SIGMA*CONC*PATHLGTH SUMMED OVER SMK CLOUDS
YIELDING SENSR.INPUT        ''ATTENUATED TGT SIGNATURE AT SENSOR LOCATION
''ATTENUATES TARGET SIGNATURE TO ACCOUNT FOR TRANSMISSION THROUGH
''ATMOSPHERE AND SMOKE CLOUDS TO SENSOR LOCATION
''
DEFINE  SPECTRUM AS AN INTEGER VARIABLE
DEFINE SIGNATURE, RANGE, SMK.ATTEN, SIG.ATMOS, ATTEN.FACT,
     SENSR.INPUT AS REAL VARIABLES
''LOOK UP APPROPRIATE SIGMA COEFF AS FUNCTION OF SENSOR WAVELENGTH
LET SIG.ATMOS = ATMOS.ATTEN(SPECTRUM)
LET ATTEN.FACT = SMK.ATTEN + SIG.ATMOS*RANGE/1000.0
IF ATTEN.FACT GT 174.0
     LET SENSR.INPUT = 0.0
     RETURN
OTHERWISE
IF SPECTRUM GT 1.5              ''THERMAL SPECTRA
     LET SENSR.INPUT = SIGNATURE*EXP.F(-ATTEN.FACT)
ELSE                            ''OPTICAL SPECTRA
     LET SENSR.INPUT = SIGNATURE/(1.0 + SKY.GROUND*(EXP.F(ATTEN.FACT)-1.0))
ALWAYS
RETURN
END
```

FIGURE II-4   ROUTINE ATTENUATION

4. Routine RESOLUTION. The RESOLUTION routine uses sensor minimum resolvable temperature (MRT) or minimum resolvable contrast (MRC) curves to convert the attenuated target signature SENSR.INPUT to a resolvable spatial frequency. The routine involves several special areas depending on the parameters which influence the MRC/MRT for each device.

GIVEN ARGUMENTS

SENSOR	INTEGER	Sensor code
MODE	INTEGER	Sensor mode of use code
INP	REAL	Attenuated target signature sensor input

YIELDING ARGUMENTS

SPATIAL.FREQ	REAL	Spatial frequency F

LOCAL VARIABLES

DEV	INTEGER	NVL device class code
NLL	INTEGER	Numbers of light levels in data array
LL	INTEGER	Light Level code
ID	INTEGER	Array index for device
IM	INTEGER	Array index for mode
I,J,N	INTEGER	Miscellaneous loop indices
AL	REAL	Ambient Light level in foot candles
C1	REAL	Coefficients for MRC/MRT curves
C2	REAL	
C3	REAL	
C4	REAL	

GLOBAL VARIABLES

SNSR.PARS	REAL 3-D	Sensor Parameters
LIGHT.LEVEL	REAL	Ambient Light Level
OPTIC.MRC	REAL 3-D	MRC definitions for DEV = 1,2,16
IL	INTEGER 1-D	Indices of light levels bracketing actual light level
SPF	REAL 1-D	Spatial frequencies at light levels in array IL.
II.MRC	REAL 3-D	MRC definitions for DEV = 3,4,5
VIDI.MRC	REAL 3-D	MRC definition for device 15
MRC.MRT	REAL 4-D	MRC definitions for devices 6,7,8,9,10,11 12,14,17,18

ENTITY ATTRIBUTES

NONE

ROUTINE AND FUNCTIONS CALLED

NONE

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-5

COMMENTARY

The RESOLUTION routine consists of four distinct computational procedures for different NVL device classes. The four procedures are different primarily because different parameters affect the MRC/MRT curves for different device classes. For example, ambient light level is vitally important for optical devices, but of no consequence for thermal devices. The line commentary for

each of the four procedures will be introduced by a short description of the factors which influence the MRC/MRT for devices in that category.

Lines 1 - 12 Declare the routine and its variables.

Lines 13 - 22 Check for valid input conditions and branch to one of the four procedure sections by NVL device code.

The first procedure block, covering devices 1, 2, and 16 handles optical MRC's. A single MRC covers all three device classes, but the MRC is a function of the light level. The data tables contain MRC coefficients for a number of discrete light levels. The MRC for the actual light level is linearly interpolated between those in the tables.

Lines 29 - 43 Find the tabled discrete light levels which bracket the actual light level.

Lines 44 - 67 Compute the spatial frequency at each of the 2 bracketing light levels. For a given light level, the MRC curve may be given in several segments for different ranges of sensor input.

Lines 68 - 71 Interplate the spatial frequncy between the two bracketing values.

The second procedure block, covering devices 3,4, and 5, handles image intensifier MRC's. Each device has it's own MRC which varies by light level and has only a single segment for each level. Interpolation on light levels is again used.

Lines 78 - 89 Select the appropriate bracketing light levels.

Lines 90 - 101 Compute the MRC for each bracketing light level.

Lines 102 - 104 Interpolate to yield the final spatial frequency.

The third procedure block covers only NVL device 15. The MRC is again given for several discrete light level bands, but no interpolation is done. Each MRC may have several segments.

Lines 108 - 119 Select the appropriate light level range.

Lines 120 - 127 Select the appropriate MRC segment and compute spatial frequency.

The fourth procedure block, encompassing device codes 6,7,8,9,10,11,12, 14,17,18 includes all devices whose MRC/MRT is given only as a function of the sensor input (perhaps in several ranges). Provision is also made for entering a different MRC/MRT curve by sensor mode although usually a single MRC/MRT will be used with only the FOV and magnification varying by mode.

Lines 136 - 156 Do preliminary checks for some devices and select the proper mode index.

Lines 157 - 167 Select the proper input range segment and compute the spatial frequency.

Finally, all four procedure blocks rejoin to account for the system magnification factor in lines 173 - 175.

```
ROUTINE RESOLUTION
''*****************
GIVEN     SENSOR,          ''SENSOR CODE
          MODE,            ''MODE OF USE CODE
          INP              ''INPUT TO SENSOR - OPTICAL CONTRAST OR TEMP DIFF
YIELDING  SPATIAL.FREQ  ''SPATIAL FREQUENCY IN CYCLES PER MILLIRADIAN
'' USES MRC OR MRT CURVE FOR GIVEN SENSOR TO CONVERT SENSOR INPUT TO
''   SPATIAL FREQUENCY FOR NVL DETECTION MODEL.
DEFINE SENSOR,MODE,DEV,NLL,LL,I,J,N,ID,IM
    AS INTEGER VARIABLES
DEFINE INP, SPATIAL.FREQ, AL, C1, C2, C3, C4
    AS REAL VARIABLES
IF INP LE 0.0
    GO TO 'RETZERO'
OTHERWISE
LET DEV = SNSR.PARS(SENSOR,MODE,9)    ''NVL DEVICE CODE NUMBER
IF (DEV LT 1) OR (DEV GT 18) OR (DEV EQ 13)
    PRINT 1 LINE WITH SENSOR, MODE, DEV, TIME.V THUS
XXXXX ERROR IN RESOLUTION  SENSOR****  MODE****  DEVICE****  TIME*****.**
    GO TO 'RETZERO'
OTHERWISE
GO TO 'DEVICE(DEV)'
''
''OPTICAL DEVICES -- NVL CODES 1,2,16 -- ONE MRC FOR ALL DEVICES WITH
''    INTERPOLATION ON LIGHT LEVELS.
'DEVICE(1)'
'DEVICE(2)'
'DEVICE(16)'
LET AL = LIGHT.LEVEL * SNSR.PARS(SENSOR,MODE,7)   ''OPTICAL GAIN
LET NLL = DIM.F(OPTIC.MRC(*))     ''NUMBER OF LIGHT LEVELS
IF AL GT OPTIC.MRC(1,1,1)     ''GREATEST LIGHT LEVEL IN TABLE
    LET IL(1) = 1   LET IL(2) = 1   LET AL = OPTIC.MRC(1,1,1)
ELSE
    IF AL LT OPTIC.MRC(NLL,1,1)    ''SMALLEST LIGHT LEVEL IN TABLE
        GO TO 'RETZERO'
    OTHERWISE  ''IN RANGE - BRACKET LIGHT LEVEL
    FOR LL = 1 TO NLL-1 DO
        IF (AL LE OPTIC.MRC(LL,1,1)) AND (AL GE OPTIC.MRC(LL+1,1,1))
            LEAVE  ''THE LOOP
        OTHERWISE
    LOOP
    LET IL(1) = LL   LET IL(2) = LL + 1  ''INDICES OF BRACKETING LIGHT LEVELS
ALWAYS
''COMPUTE SPATIAL FREQUENCY FOR EACH BRACKETING LIGHT LEVEL
FOR I = 1 TO 2 DO
    LET LL = IL(I)
```

FIGURE II-5   ROUTINE RESOLUTION

```
        LET SPF(I) = 0.0
        LET N = DIM.F(OPTIC.MRC(LL,*,*))
        IF INP GT OPTIC.MRC(LL,N,2)   ''MAX INPUT
            LET SPF(I) = OPTIC.MRC(LL,1,2)    ''MAX FREQ
            CYCLE
        OTHERWISE
        FOR J = 2 TO N DO    ''LOOP OVER ALL SEGMENTS
            IF (INP GE OPTIC.MRC(LL,J,1)) AND (INP LE OPTIC.MRC(LL,J,2))
                LET C1 = OPTIC.MRC(LL,J,3)
                LET C2 = OPTIC.MRC(LL,J,4)
                LET C3 = OPTIC.MRC(LL,J,5)
                LET C4 = OPTIC.MRC(LL,J,6)
                IF OPTIC.MRC(LL,J,7) EQ 1
                    LET SPF(I) = (C1 + C2*INP)/(C3 + C4*INP)
                ELSE
                    LET SPF(I) = C1 * (INP ** C2)
                ALWAYS
                LEAVE  ''THE SEGMENT LOOP
            OTHERWISE
        LOOP
    LOOP
    ''INTERPOLATE BETWEEN THE BRACKETED FREQUENCIES
    LET SPATIAL.FREQ = SPF(2) + (SPF(1)-SPF(2)) *
        (AL - OPTIC.MRC(IL(2),1,1)) / (OPTIC.MRC(IL(1),1,1) - OPTIC.MRC(IL(2),1,1))
    GO TO 'OUT'
    ''
    ''IMAGE INTENSIFIERS -- NVL DEVICES 3,4,5 -- MRC FOR EACH DEVICE AND
    ''      INTERPOLATE OVER LIGHT LEVELS
    'DEVICE(3)'
    'DEVICE(4)'
    'DEVICE(5)'
    LET AL = LIGHT.LEVEL
    LET ID = DEV - 2
    LET NLL = DIM.F(II.MRC(ID,*,*))    ''NUMBER OF LIGHT LEVELS
    IF (AL LT II.MRC(ID,NLL,1)) OR (AL GT II.MRC(ID,1,1))
        GO TO 'RETZERO'
    OTHERWISE
    FOR LL = 1 TO NLL-1 DO
        IF (AL LE II.MRC(ID,LL,1)) AND (AL GE II.MRC(ID,LL+1,1))
            LEAVE   ''THE LOOP
        OTHERWISE
    LOOP
    LET IL(1) = LL    LET IL(2) = LL+1
    FOR I = 1 TO 2 DO
        LET SPF(I) = 0.0
        LET LL = IL(I)
```

FIGURE II-5   (CONTINUED)

```
    IF INP LT II.MRC(ID,LL,2)
        CYCLE
    OTHERWISE
    LET C1 = II.MRC(ID,LL,3)
    LET C2 = II.MRC(ID,LL,4)
    LET C3 = II.MRC(ID,LL,5)
    LET C4 = II.MRC(ID,LL,6)
    LET SPF(I) = (C1 + C2 * INP) / (C3 + C4 * INP)
LOOP
LET SPATIAL.FREQ = SPF(2) + (SPF(1) - SPF(2)) *
    (AL - II.MRC(ID,IL(2),1)) / (II.MRC(ID,IL(1),1) - II.MRC(ID,IL(2),1))
GO TO 'OUT'
''
''VIDICON -- NVL DEVICE 15 -- MRC DEPENDS ON LIGHT LEVEL, BUT NO INTERPOLATION
'DEVICE(15)'
LET AL = LIGHT.LEVEL
LET NLL = DIM.F(VIDI.MRC(*))
IF AL GT VIDI.MRC(1,1,2)        ''MAX LIGHT LEVEL
    LET AL = VIDI.MRC(1,1,2)
ALWAYS
IF AL LT VIDI.MRC(NLL,1,1)      ''MIN LIGHT LEVEL
    GO TO 'RETZERO'
OTHERWISE
FOR LL = 1 TO NLL DO
    IF (AL LT VIDI.MRC(LL,1,1)) OR (AL GT VIDI.MRC(LL,1,2))
        CYCLE
    OTHERWISE
    LET N = DIM.F(VIDI.MRC(LL,*,*))
    FOR J = 2 TO N DO
        IF (INP GE VIDI.MRC(LL,J,1)) AND (INP LE VIDI.MRC(LL,J,2))
            LET C1 = VIDI.MRC(LL,J,3)
            LET C2 = VIDI.MRC(LL,J,4)
            LET C3 = VIDI.MRC(LL,J,5)
            LET C4 = VIDI.MRC(LL,J,6)
            LET SPATIAL.FREQ = (C1 + C2 * INP) / (C3 + C4 * INP)
            LEAVE   ''THE SEGMENT LOOP
        OTHERWISE
    LOOP
LOOP
GO TO 'OUT'
''
''DEVICES WHOSE MRC/MRT DOES NOT DEPEND ON LIGHT LEVELS:
''SILICON TV -- NVL DEVICES 10, 12
'DEVICE(10)'
'DEVICE(12)'
LET AI = LIGHT.LEVEL
```

FIGURE II-5   (CONTINUED)

```
IF AL LT 92.9
     GO TO 'RETZERO'
OTHERWISE
''
''THERMAL DEVICES -- NVL DEVICES 6,7,8,9,11,14,17,18
'DEVICE(6)'
'DEVICE(7)'
'DEVICE(8)'
'DEVICE(9)'
'DEVICE(11)'
'DEVICE(14)'
'DEVICE(17)'
'DEVICE(18)'
IF MODE EQ 2 AND DIM.F(MRC.MRT(DEV,*,*,*)) EQ 2
     LET IM = 2
ELSE
     LET IM = 1
ALWAYS
LET N = DIM.F(MRC.MRT(DEV,IM,*,*))
FOR J = 1 TO N DO
     IF (INP GE MRC.MRT(DEV,IM,J,1)) AND (INP LE MRC.MRT(DEV,IM,J,2))
          LET C1 = MRC.MRT(DEV,IM,J,3)
          LET C2 = MRC.MRT(DEV,IM,J,4)
          LET C3 = MRC.MRT(DEV,IM,J,5)
          LET C4 = MRC.MRT(DEV,IM,J,6)
          LET SPATIAL.FREQ = (C1 + C2 * INP) / (C3 + C4 * INP)
          LEAVE  ''THE LOOP
     OTHERWISE
LOOP
GO TO 'OUT'
''
'RETZERO'
LET SPATIAL.FREQ = 0.0
''
'OUT'
LET SPATIAL.FREQ = SPATIAL.FREQ * SNSR.PARS(SENSOR,MODE,6)   ''MAGNIFICATION
RETURN
END
```

FIGURE II-5 (CONTINUED)

5. <u>Routine JOHNSON.CRITERION</u>. This routine looks up the Johnson n50 criterion to be used in the NVL model. The routine is a simple table lookup as a function of device, acquisition level, and whether the target is stationary or moving.

GIVEN ARGUMENTS

DEVICE	INTEGER	NVL device code
ACQ.LEV	INTEGER	Required acquisition level code
TGT	INTEGER	Pointer to target entity

YIELDING ARGUMENT

N50	REAL	Number of cycles for 50% probability

LOCAL VARIABLES

NONE

GLOBAL VARIABLES

N50TABLE	REAL 3-D	Array of n50 values

ENTITY ATTRIBUTES

SPD	REAL	Target speed

ROUTINES AND FUNCTIONS CALLED

NONE

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-6.

COMMENTARY (JOHNSON.CRITERION)

Code is Self-Explanatory

```
ROUTINE JOHNSON.CRITERION
''************************
GIVEN     DEVICE,   ''NVL DEVICE CLASS
          ACQ.LEV,  ''REQUIRED ACQUISITION LEVEL CODE
          TGT       ''POINTER TO TARGET ENTITY
YIELDING N50        ''NUMBER CYCLES FOR 50% ACQUISITION PROB.
''LOOKS UP JOHNSON CRITERION LEVEL TO BE USED IN NVL DETECTION MODEL
DEFINE ACQ.LEV, DEVICE, TGT AS INTEGER VARIABLES
DEFINE N50 AS A REAL VARIABLE
IF SPD(TGT) LT 0.01
    LET N50 = N50TABLE(DEVICE,ACQ.LEV,1)     ''STATIONARY TARGET
ELSE
    LET N50 = N50TABLE(DEVICE,ACQ.LEV,2)     ''MOVING TARGET
ALWAYS
RETURN
END
```

FIGURE II-6 ROUTINE JOHNSON.CRITERION

6. Routine PR.INFINITY. The PR.INFINITY routine computes the infinite-time probability of acquisition, P_∞, by applying the target transform probability function to the N/n50 ratio.

GIVEN ARGUMENT

CYCLE.RATIO	REAL	N/n50 RATIO

YIELDING ARGUMENT

PROB.INF	REAL	P_∞ PROBABILITY OF ACQUISITION

LOCAL VARIABLES

CR	REAL	CYCLE.RATIO SCALED

GLOBAL VARIABLES

NONE

ENTITY ARRTIBUTES

NONE

ROUTINES AND FUNCTIONS CALLED

NONE

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-7.

COMMENTARY (PR.INFINITY

The PR.INFINITY Routine approximates a cumulative normal probability function.

```
ROUTINE PR.INFINITY
''**********************
GIVEN     CYCS,                ''RESOLVABLE CYCLES ON TARGET
          N50                  ''JOHNSON CRITERION CYCLES FOR 50% PROB
YIELDING PROB.INF              ''PROBABILITY OF ACQUISITION GIVEN INFINITE TIME
'' P INFINITY CALCULATION -- NORMAL CDF WITH MU = N50, AND SIGMA = 0.632*MU
DEFINE CYCS,N50,X,AX,P,AXSQ,PROB.INF AS REAL VARIABLES
LET X = (CYCS - N50)/(0.632 * N50)   ''STANDARDIZE
IF X LT -10.0
    LET PROB.INF = 0.0
ELSE
    IF X GT 10.0
        LET PROB.INF = 1.0
    ELSE  ''USE POLYNOMIAL APPROX 26.2.18 FROM ABRAMOWITZ
        LET AX = ABS.F(X)
        LET AXSQ = AX * AX
        LET P = 1.0 + 0.196854*AX + 0.115194*AXSQ + 0.000344*AX*AXSQ +
                0.019527*AXSQ*AXSQ
        LET PROB.INF = 1.0 - 1.0 / (2.0 * P **4)
        IF X LT 0.0
            LET PROB.INF = 1.0 - PROB.INF
        ALWAYS
    ALWAYS
ALWAYS
RETURN
END
```

FIGURE II-7   ROUTINE PR.INFINITY

7. <u>Routine SCH.RATE</u>. The SCH.RATE routine applies the Lawson Model to compute the search rate $1/\tau_1$ for the NVL methodology.

GIVEN ARGUMENTS

SENSOR	INTEGER	SENSOR Code
MODE	INTEGER	SENSOR Mode of use code
NBYN50	REAL	N/n50 cycle ratio
HFOS	REAL	Angle of observer's horizontal field of search

YIELDING ARGUMENT

RATE	REAL	$1/\tau_1$ Search rate

LOCAL VARIABLES

TO	REAL	Time to search one sensor device screen
HFOV	REAL	Angle of device horizontal field of view
VFOV	REAL	Angle of device vertical field of view
TS	REAL	Time to search entire field of search
PS	REAL	Conditional detection probability for one FOV

GLOBAL VARIABLES

SNSR.PARS	REAL 3-D	NVL SENSOR Device Parameters

ENTITY ATTRIBUTES

NONE

ROUTINES AND FUNCTIONS CALLED

EXP.F

EVENTS SCHEDULED

NONE

SIMSCRIPT CODE

See Figure II-8.

COMMENTARY (SCH.RATE)

See VOLUME I, CHAPTER III, SECTION A for discussion of the LAWSON SEARCH MODEL EQUATIONS.

```
ROUTINE SCH.RATE(SENSOR,MODE,NBYN50,HPOS,VPOS) YIELDING RATE
''*****************************************************************
DEFINE SENSOR, MODE  AS INTEGER VARIABLES
DEFINE NBYN50, RATE, T0, PS, TS, HPOS, VPOS, HPOV, VPOV AS REAL VARIABLES
LET T0 = 1.7
LET HPOV = SNSR.PARS(SENSOR,MODE,4)
LET VPOV = SNSR.PARS(SENSOR,MODE,5)
IF VPOS GT VPOV
    LET TS = (T0 * HPOS * VPOS)/(HPOV * VPOV)
ELSE
    LET TS = (T0 * HPOS) / HPOV
ALWAYS
LET PS = 1.0 - EXP.F(-NBYN50*T0/6.4)
LET RATE = (2.0 * PS) / (TS * (2.0 - PS))
RETURN
END
```

FIGURE II-8   ROUTINE SCH.RATE

D. SUPPORTING DATA ARRAYS

The STAR implementation of the NVL methodology uses several global data arrays. These have been listed under the various routines in the preceeding sections and will be discussed in detail here. All of these arrays are initialized by Routine RES.SCH which is included as Figure II-9.

1. SNSR.PARS (SENSOR, MODE, J). The SNSR.PARS array is a 3-dimensional real array indexed by

SENSOR	-	SENSOR CODE
MODE	-	SENSOR MODE of use Code
J	-	(Third Index)

For a given SENSOR and MODE, the array contains the following sensor parameters:

J = 1	Sensor maximum acquisition range
J = 2	Minimum sensor input threshold
J = 3	Wavelength spectrum code
J = 4	Horizontal field of view
J = 5	Vertical field of view
J = 6	Magnification level
J = 7	Optical gain
J = 8	Range behind TGT within which to check background type
J = 9	NVL device code

2. MRC.MRT (SENSOR, MODE, SEG, J). The MRC.MRT is a 4-dimensional real array which contains coefficients defining the sensor MRC and MRT curves for most of the NVL devices. The array is indexed by

DEV	-	NVL Device Code
MODE	-	SENSOR Mode of use Code
SEG	-	Segment number - each curve can be divided into as many segments as the user wishes to use.
J	-	(Fourth Index)

For a given SENSOR, MODE, and SEG the MRC.MRT array contains the following parameters:

J = 1	Lower bound on SENSOR input for this segment
J = 2	Upper bound on SENSOR input for this segment
J = 3	Coefficient C_1
J = 4	Coefficient C_2
J = 5	Coefficient C_3
J = 6	Coefficient C_4

Within each segment the MRC/MRT curve is defined as the linear fractional function

$$f = \frac{C_1 + C_2 * X}{C_3 + C_4 * X}$$

were X is the sensor input.

In addition to the MRC.MRT array, three other arrays contain device MRC/MRT curve parameters:

3. <u>OPTIC.MRC (LL, SEG, K)</u>. The OPTIC.MRC array is a 3-dimensional real array containing optical device MRC's indexed by:

LL	-	Light Level Code
SEG	-	Segment Number
K	-	(Third Index)

The light levels are assumed to be entered in order of decreasing illumination level - brightest first. For each light level index LL, the OPTIC.MRC array contains the following parameters:

For SEG = 1 the array contains bound information

OPTIC.MRC (LL, 1, 1) = light level in foot candles

OPTIC.MRC (LL, 1, 2) = upper bound on spatial frequency at this light level.

The remaining values SEG = 2, ..., N index N-1 segments of an MRC curve. For a given LL and SEG $\geq$ 2 the arraycontains the following:

K = 1 lower bound on sensor input for this segment

K = 2 upper bound on sensor input for this segment

K = 3 Coefficient C_1

K = 4 Coefficient C_2

K = 5 Coefficient C_3

K = 6 Coefficient C_4

K = 7 Function type code.

If the function type code = 1, then the MRC functional form is the linear fractional form given above. If the function type is 2, then

F = C_1 * (X**C2).

4. <u>II.MRC (DEV, LL, K)</u>. The II.MRC array is a 3-dimensional real array of image intensifier MRC coefficients indexed by:

DEV	-	NVL device code minus 2 (device codes 3, 4, 5 yield array indices 1, 2, 3)
LL	-	light level code
K	-	(Third Index)

The light levels are assumed to be arranged in order of decreasing illumination - brightest first. For each given DEV and LL, the array contains:

K = 1 Light level in foot candles

K = 2 Minimum sensor input threshold

K = 3 Coefficient C_1

K = 4 Coefficient C_2

K = 5 Coefficient C_3

K = 6 Coefficient C_4

The linear fractional functional form is assumed.

5. VIDI.MRC (LL, SEG, K). The VIDI.MRC array is a 3-dimensional real array containing vidicon MRC coefficients indexed by

LL - Light level band code

SEG - MRC segment number

K - (Third Index)

The light level bands are assumed to be arranged in order of decreasing illumination - brightest first. For a given LL, the VIDI.MRC array contains the following parameters:

For SEG = 1 the array contains the light level band bounds:

VIDI.MRC (LL, 1, 1) = lower bound on light level for this band

VIDI.MRC (LL, 1, 2) = upper bound on light level for this band

The remaining values SEG = 2, ..., N index N-1 segments of an MRC curve. For a given LL and SEG $\geq$ 2, the array contains the following:

K = 1 lower bound on sensor input for this segment

K = 2 upper bound on sensor input for this segment

K = 3   Cofficient C_1

K = 4   Cofficient C_2

K = 5   Cofficient C_3

K = 6   Cofficient C_4

The MRC is assumed to follow the linear fractional form.

6. TAR.SIG (SYS, WPN, SPECTRUM, BACKGROUND). The TAR.SIG array is a 4-dimensional real array of target signature values indexed by

SYS	-	System Type of Target Entity
WPN	-	Weapon Type of Target Entity
SPECTRUM	-	SENSOR Wavelength Code
BACKGRND	-	Target Background Type Code

7. MX.SIG (SYS, WPN, SPECTRUM.) The MX.SIG array is a 3-dimensional real array which contains optimistic, background - independent target signature values. The value of MX.SIG (SYS, WPN, SPECTRUM) is computed by the RES.SCH routine to be the maximum over all background codes of TAR.SIG (SYS, WPN, SPECTRUM, BACKGROUND).

8. ATMOS.ATTEN (SPECTRUM). The ATMOS.ATTEN array is a 1-dimensional real array of attenuation coefficients indexed by

SPECTRUM - Wavelength Band Code

For each SPECTRUM, the array contains the attenuation coefficient for open air (for whatever background meteorological conditions are assumed for the simulated battle).

9. N50TABL (DEVICE, ACQ.LEV, MOVE). The N50TABLE array contains Johnson Criterion values in a 3-dimensional real array indexed by

DEVICE	-	NVL Device Code
ACQ.LEV	-	Acquisition Level Code
MOVE	-	1 = Stationary Target, 2 = Moving Target

The input value sequence for initializing these arrays is found in Chapter V of Volume I of this report. It can also be inferred from the code for routine RES.SCH which is given in Figure II-9.

```
ROUTINE RES.SCH
''*********************************************************************
'' READS IN DATA FOR ALL SEARCH TYPES AND DATA FOR NVL SEARCH MODEL
DEFINE MXTYP, NTYPS, TYPE, NOBS, J, TAC, NPARS, K, NSENS, SEN, MOD,
    NMODES, N, I, SYS, WPN, MXSEN, MXMODE,L,NSEGS,SPECTRUM,NLL
    AS INTEGER VARIABLES
DEFINE MSIG AS A REAL VARIABLE
'' READ DATA TO DEFINE SEARCH TYPES AND SEARCH TACTIC PARAMETERS
SKIP.WORDS
RESERVE TYPE.SCH AS  MXSYS BY MXWPN
FOR I = 1 TO NUMBER.OF.SYSTEMS, DO
   READ J,L,  TYPE.SCH(J,L)
   PRINT 1 LINE WITH J,L,TYPE.SCH(J,L) THUS
J=***  L=***  TYPE.SCH=****
LOOP
READ MXTYP, NTYPS
PRINT 1 LINE WITH MXTYP,NTYPS THUS
MXTYP ****     NTYPS *****
RESERVE SCH.DATA (*,*,*) AS MXTYP BY *
FOR I = 1 TO NTYPS, DO
    READ TYPE, NOBS
    PRINT 1 LINE WITH TYPE,NOBS THUS
TYPE **** NOBS ****
    RESERVE SCH.DATA(TYPE,*,*) AS NOBS BY *
    FOR J = 1 TO NOBS, DO
         READ TAC, NPARS
         PRINT 1 LINE WITH TAC,NPARS THUS
TAC **** NPARS ****
         RESERVE SCH.DATA(TYPE,J,*) AS NPARS    ''NOTE NPARS INCLUDES TAC
         LET SCH.DATA(TYPE,J,1) = TAC          ''WHICH HAS BEEN READ ALREADY
         PRINT 1 LINE WITH SCH.DATA(TYPE,J,1) THUS
SCH.DATA1 = *****.****
         FOR K = 2 TO NPARS, DO
              READ SCH.DATA(TYPE,J,K)
              PRINT 1 LINE WITH SCH.DATA(TYPE,J,K) THUS
         SCH.DATAK ******.*****
         LOOP
    LOOP
LOOP
'' READ SENSOR PARAMETER DATA
SKIP.WORDS
READ NSENS   ''NUMBER OF SENSORS TO INCLUDE
IF NSENS EQ 0
    RETURN   '' NVL MODEL NOT BEING USED, SO SKIP THE REST
OTHERWISE
READ MXSEN
```

FIGURE II-9   ROUTINE RES.SCH

```
RESERVE SNSR.PARS(*,*,*) AS MXSEN BY *
FOR I = 1 TO NSENS DO
    READ SEN, MXMODE, NMODES
    RESERVE SNSR.PARS(SEN,*,*) AS MXMODE BY 9
    FOR J = 1 TO NMODES DO
        READ MOD
        FOR K = 1 TO 9 READ SNSR.PARS(SEN,MOD,K)
    LOOP
LOOP
RESERVE IL(*), SPF(*) AS 2
''INPUT NVL DEVICE DATA -- MRC/MRT CURVES
''OPTICAL DEVICES
SKIP.WORDS
READ NLL              ''NUMBER OF LIGHT LEVELS
RESERVE OPTIC.MRC(*,*,*) AS NLL BY *
FOR I = 1 TO NLL DO
    READ NSEGS        ''NUMBER OF MRC SEGMENTS
    RESERVE OPTIC.MRC(I,*,*) AS NSEGS+1 BY *
    RESERVE OPTIC.MRC(I,1,*) AS 2
    READ OPTIC.MRC(I,1,1), OPTIC.MRC(I,1,2)
    FOR J = 2 TO NSEGS+1 DO
        RESERVE OPTIC.MRC(I,J,*) AS 7
        FOR K = 1 TO 7 READ OPTIC.MRC(I,J,K)
    LOOP
LOOP
''IMAGE INTENSIFIERS
SKIP.WORDS
READ N                ''NUMBER OF II DEVICES
READ NLL              ''NUMBER OF LIGHT LEVELS
RESERVE II.MRC(*,*,*) AS N BY NLL BY 6
FOR I = 1 TO N
FOR J = 1 TO NLL
FOR K = 1 TO 6 READ II.MRC(I,J,K)
''THERMAL DEVICES AND SILICON TV
SKIP.WORDS
READ MXTYP            ''LARGEST NVL DEVICE CODE
READ NTYPS            ''NUMBER OF DEVICES USING MRC.MRT TABLE
RESERVE MRC.MRT(*,*,*,*) AS MXTYP BY *
FOR I = 1 TO NTYPS DO
    READ SEN          ''NVL DEVICE NUMBER
    READ NMODES       ''NUMBER OF MODES WHICH HAVE DISTINCT MRC/MRT CURVES
    RESERVE MRC.MRT(SEN,*,*,*) AS NMODES BY *
    FOR J = 1 TO NMODES DO
        READ MOD,NSEGS
        RESERVE MRC.MRT(SEN,MOD,*,*) AS NSEGS BY 6
        FOR K = 1 TO NSEGS  FOR L = 1 TO 6 READ MRC.MRT(SEN,MOD,K,L)
```

FIGURE II-9   (CONTINUED)

```
        LOOP
    LOOP
    ''VIDICON
    SKIP.WORDS
    READ NLL              ''NUMBER OF LIGHT LEVELS
    RESERVE VIDI.MRC(*,*,*) AS NLL BY *
    FOR I = 1 TO NLL DO
        READ L,NSEGS
        RESERVE VIDI.MRC(L,*,*) AS NSEGS+1 BY *
        RESERVE VIDI.MRC(L,1,*) AS 2
        READ VIDI.MRC(L,1,1) , VIDI.MRC(L,1,2)
        FOR J = 2 TO NSEGS+1 DO
            RESERVE VIDI.MRC(L,J,*) AS 6
            FOR K = 1 TO 6 READ VIDI.MRC(L,J,K)
        LOOP
    LOOP
    '' READ TARGET SIGNATURE DATA
    SKIP.WORDS
    READ N           ''NUMBER OF SPECTRAL BANDS
    READ MXBKGND     ''NUMBER OF BACKGROUND CONDITIONS
    RESERVE TAR.SIG(*,*,*,*) AS MXSYS BY MXWPN  BY *
    RESERVE MX.SIG(*,*,*) AS MXSYS BY MXWPN  BY *
    FOR I = 1 TO NUMBER.OF.SYSTEMS DO
        READ SYS, WPN
        RESERVE TAR.SIG(SYS,WPN,*,*) AS N BY MXBKGND
        RESERVE MX.SIG(SYS,WPN,*) AS N
        FOR J = 1 TO N DO
            READ SPECTRUM
            LET MSIG = -1000.0
            FOR K = 1 TO MXBKGND DO
                READ TAR.SIG(SYS,WPN,SPECTRUM,K)
                IF TAR.SIG(SYS,WPN,SPECTRUM,K) GT MSIG
                    LET MSIG = TAR.SIG(SYS,WPN,SPECTRUM,K)
                ALWAYS
            LOOP
            LET MX.SIG(SYS,WPN,SPECTRUM) = MSIG
        LOOP
    LOOP
    '' READ ATMOSPHERIC ATTENUATION DATA
    SKIP.WORDS
    READ SKY.GROUND     ''SKY/GROUND RATIO
    READ LIGHT.LEVEL    ''IN FOOT CANDLES
    READ N  ''NUMBER OF SPECTRUM BANDS
    RESERVE ATMOS.ATTEN(*) AS N
    FOR I = 1 TO N
        READ ATMOS.ATTEN(I)
```

FIGURE II-9 (CONTINUED)

```
'' READ JOHNSON CRITERION DATA
SKIP.WORDS
READ MXTYP, NTYPS
RESERVE N50TABLE(*,*,*) AS MXTYP BY *
FOR I = 1 TO NTYPS DO
    READ SEN          ''NVL DEVICE CLASS
    RESERVE N50TABLE(SEN,*,*) AS 5 BY 2          ''K=1 STAT TGT,  K=2 MV TGT
    FOR K = 1 TO 2
    FOR J = 1 TO 5  READ N50TABLE(SEN,J,K)
LOOP
RETURN
END
```

FIGURE II-9 (CONTINUED)

E. LINE OF SIGHT BACKGROUND COMPUTATIONS.

1. Definition of Background Codes. The NVL detection model uses a target signature which involves comparing the target with its background as perceived from the observer's location. The nature of the background can have a dramatic effect on the difficulty of acquiring the target (as an example imagine detection of a helicopter against a cluttered background of terrain and trees as compared with detection when the background is open sky). The STAR line of sight model has been adapted to compute target backgrounds. The resulting routine is called LOS.BKGRND and has the calling sequence:

CALL LOS.BKGND (MAXDIST) YIELDING BKGND, DISTOBKG where:

MAXDIST	REAL	Input parameter giving the distance beyond the target within which detailed background computations are to be done.
BKGND	INTEGER	Return code which indicates the background type.
DISTOBKG	REAL	Return value giving the distance to the point at which the background was determined. This distance is a rough estimate of the distance beyond the target to the background.

The background codes which are returned by the routine allow target signature data to be used at several levels of resolution. The interpretation of these background codes depends on the number, MXBKGND, of background types included in the input target signature data.

If MXBKGND = 1, then all background checks in the simulation are skipped, and the single set of target signature data is always used regardless of the situation.

If MXBKGND = 2, then the program distinguishes between terrain (return Code = 1) and sky (Code = 2) as background types.

If MXBKGND = 3, then return Codes 1 and 2 are as in the MXBKGND = 2 case. In addition, the program will check for smoke clouds between the target and its background. If smoke is found to be the background, then the return Code = 3.

If MXBKGND = 4, then the program distinguishes several terrain types. The return codes in this case have the following meaning:

1: Terrain of undesignated type more than MAXDIST behind the target

2: Sky

3: Hills within MAXDIST beyond target

4: Trees within MAXDIST beyond target

Finally, if the user inputs 5 sets of target signature data, then the program adds the smoke background checks to the above four codes, and if smoke is found to be the background, the return code is set to 5.

2. LOS.BKGND Routine. The LOS.BKGND routine follows the same basic structure as the STAR routine LOS. (Reference 3) It is intended that a call to LOS.BKGND will always be immediately preceded by a LOS call for the same observer/target pair. LOS.BKGND shares with LOS an extensive set of global variables (all carrying the suffix .LS) and expects that the preceding call to LOS has set up values for several of these variables which define the basic locations and posture of the observer (denoted as Element A) and the target (denoted as Element B). LOS.BKGND should not be called if line of sight from A to B does not exist (as determined by LOS) since then the concept of background is meaningless.

The LOS.BKGND SIMSCRIPT code appears in Figure 11-10. In the following commentary we briefly discuss the function of various sections of the code.

Lines 1 - 12 Declare the routine and its local variables.

Lines 13 - 46 Extend the observer/target line through the center of the exposed portion of the target, beyond the target by a distance MAXDIST. For the remainder of the routine the target is denoted as Point B, while point A refers to this newly defined point MAXDIST beyond the target.

The routine now concentrates on the line segment between A and B. As in the LOS routine we parameterize this line using a single variable S such that

X(S) = XA + S* (XB-XA)

Y(S) = YA + S* (YB-YA)

Z(S) = ZA + S* (ZB-ZA)

so that S = 0 corresponds to Point A and S = 1 corresponds to the target at Point B.

We are hunting for obstruction to the line of sight between A and B, and the obstruction with the largest value of S ($0 \leq S \leq 1$) defines the background feature closest to the target (B) and thus defines the BKGND code and the distance DISTOBKG. Obstructions can be of two types: terrain hills or forest features, so the routine will have to consider all hills and forest features which might intersect the line between B (the target) and A (the far end of the O/T line extended beyond the target).

As the routine loops through hills and forest ellipses, it updates 3 global variables

GRND.BLK.LS	=	S value corresponding to the ground obstruction closest to B which has been found so far
TRE.BLK.LS	=	S value corresponding to the tree obstruction closest to B which has been found so far
MAX.BLK.LS	=	Maximum of GRND.BLK.LS and TRE.BLK.LS

```
ROUTINE LOS.BKGND(MAXDIST) YIELDING BKGND, DISTOBKG
''*************************************************************
''COMPUTES BACKGROUND OF TGT B AS VIEWED BY OBSERVER A AND DIST TO BKGRND
''BKGND RETURN CODES ARE    1 = TERRAIN OF UNSPECIFIED TYPE (DISTANT)
''                          2 = SKY
''                          3 = HILL WITHIN MAXDIST BEYOND TARGET
''                          4 = TREES WITHIN MAXDIST BEYOND TARGET
''NOTE --- THIS ROUTINE USES THE SAME SET OF GLOBAL VARIABLES AS LOS AND
''         ASSUMES THAT SOME OF THESE HAVE ALREADY BEEN GIVEN VALUES BY
''         A PRECEDING SIGHT(A,B) CALL.
DEFINE I,K,N,L,IC,M,BKGND,BLOCKED AS INTEGER VARIABLES
DEFINE MAXDIST, DISTOBKG AS A REAL VARIABLE
IF MXBKGND EQ 1
    LET BKGND = 1
    LET DISTOBKG = MAXDIST
    RETURN
OTHERWISE
LET ZB.LS = ZB.LS - SIZEB.LS * 0.5 * VISFRB.LS    ''CTR OF VIS PART OF TGT
LET ZBA.LS = ZB.LS-ZA.LS
IF XBA.LS EQ 0.0 AND YBA.LS EQ 0.0      ''DEGENERATE CASE
    IF ZBA.LS LT 0
        LET BKGND = 1
        LET DISTOBKG = MAX.F(TMICB.LS,0.0)
    ELSE
        LET BKGND = 2
        LET DISTOBKG = MAXDIST
    ALWAYS
    RETURN
OTHERWISE
IF MAXDIST LE 1.0        '' DEGENERATE CASE
    LET DISTOBKG = MAXDIST
    IF ZBA.LS LT 0
        LET BKGND = 1
    ELSE
        LET BKGND = 2
    ALWAYS
    RETURN
OTHERWISE
LET SQ.LS = MAXDIST/SQRT.F(XBA.LS*XBA.LS + YBA.LS*YBA.LS + ZBA.LS*ZBA.LS)
''REDEFINE POINT A TO BE FAR END OF THE EXTENDED O-T LINE
LET XA.LS = XB.LS + SQ.LS * XBA.LS         LET XBA.LS = XB.LS - XA.LS
LET YA.LS = YB.LS + SQ.LS * YBA.LS         LET YBA.LS = YB.LS - YA.LS
LET ZA.LS = ZB.LS + SQ.LS * ZBA.LS         LET ZBA.LS = ZB.LS - ZA.LS
LET XBASQ.LS = XBA.LS **2         LET YBASQ.LS = YBA.LS **2
LET XYBA.LS = XBA.LS * YBA.LS
LET TWOXBA.LS = 2. * XBA.LS       LET TWOYBA.LS = 2. * YBA.LS
```

FIGURE II-10   ROUTINE LOS.BKGND

```
ADD 1 TO KTREP                       LET CHTMAX.LS = 0.
''COMPUTE LIST OF GRIDSQUARES CROSSED BY EXTENDED O-T LINE
LET NGRSQ.LS = 0
IF XBA.LS EQ 0. LET XBA.LS = 0.1 ALWAYS
IF XBA.LS GT 0.
    LET ISGX.LS = -1       LET XINC.LS = GSIZE/XBA.LS
ELSE
    LET ISGX.LS = 1        LET XINC.LS =-GSIZE/XBA.LS
ALWAYS
IF YBA.LS EQ 0. LET YBA.LS = 0.1 ALWAYS
IF YBA.LS GT 0.
    LET ISGY.LS = -1       LET YINC.LS = GSIZE/YBA.LS
ELSE
    LET ISGY.LS = 1        LET YINC.LS = -GSIZE/YBA.LS
ALWAYS
LET IX.LS = 1 + TRUNC.F((XB.LS-X.LO.BDRY)/GSIZE)
LET IY.LS = 1 + TRUNC.F((YB.LS - Y.LO.BDRY) /GSIZE)
LET XSTEP.LS = (XB.LS-X.LO.BDRY-GSIZE*(IX.LS+0.5*(ISGX.LS-1.)))/XBA.LS
LET YSTEP.LS = (YB.LS-Y.LO.BDRY-GSIZE*(IY.LS+0.5*(ISGY.LS-1.)))/YBA.LS
'GRID.LOOP'
IF (IX.LS GE 1) AND(IX.LS LE NXGRID) AND(IY.LS GE 1) AND(IY.LS LE NYGRID)
    ADD 1 TO NGRSQ.LS
    LET IGX.LS(NGRSQ.LS) = IX.LS  LET IGY.LS(NGRSQ.LS) = IY.LS
ALWAYS
IF XSTEP.LS LE 1. OR YSTEP.LS LE 1.
    IF XSTEP.LS LE YSTEP.LS
        ADD ISGX.LS TO IX.LS     ADD XINC.LS TO XSTEP.LS
    ALWAYS
    IF XSTEP.LS GE YSTEP.LS
        ADD ISGY.LS TO IY.LS     ADD YINC.LS TO YSTEP.LS
    ALWAYS
    GO TO 'GRID.LOOP'
OTHERWISE ''GRID LIST IS COMPLETE IN IGX.LS, IGY.LS WITH NGRSQ.LS ENTRIES
IF NGRSQ.LS EQ 0    '' THEN TOTALLY OFF THE MAP
    LET DISTOBKG = MAXDIST
    IF ZBA.LS LT 0
        LET BKGND = 2
    ELSE
        LET BKGND = 1
    ALWAYS
    RETURN
OTHERWISE
'' NOW FIND WHICH COVER ELLIPSES INTERSECT THE EXTENDED O-T LINE
'' AND CHECK BLOCKAGE AT S1 AND S2 FOR EACH SUCH ELLIPSE
LET TRE.BLK.LS = 0.0   LET GRND.BLK.LS = 0.0   LET MAX.BLK.LS = 0.0
LET NELS.LS = 0
```

FIGURE II-10 (CONTINUED)

```
IF NCVELS EQ 0 GO TO 'HILL.PROCESSING' OTHERWISE
FOR K = 1 TO NGRSQ.LS DO
    LET IX.LS = IGX.LS(K)      LET IY.LS = IGY.LS(K)
    LET DUM.I(*) = LISTC(IX.LS,IY.LS,*)      LET N = DUM.I(1) + 1
    IF N EQ 1 CYCLE ELSE
    FOR L = 2 TO N DO
        LET IC = DUM.I(L)      LET CPK.LS = HT.E(IC)
        IF KCREP(IC) EQ KTREP CYCLE ELSE
        LET KCREP(IC) = KTREP
        LET RX.LS = XA.LS - XC.E(IC)      LET RY.LS = YA.LS - YC.E(IC)
        LET PXX.LS = PXX.E(IC)    LET PYY.LS = PYY.E(IC)    LET PXY.LS = PXY.E(IC)
        LET AA.LS = PXX.LS*XBASQ.LS + PYY.LS*YBASQ.LS  + PXY.LS*XYBA.LS
        LET BB.LS = PXX.LS*TWOXBA.LS*RX.LS + PYY.LS*TWOYBA.LS*RY.LS
             +PXY.LS*(RX.LS*YBA.LS + RY.LS*XBA.LS)
        LET CC.LS = PXX.LS*RX.LS**2 + PYY.LS*RY.LS**2 + PXY.LS*RX.LS*RY.LS -1.0
        LET ARG.LS = BB.LS**2 - 4.0*AA.LS*CC.LS
        IF ARG.LS LE 0. CYCLE ELSE
        LET SQ.LS = SQRT.F(ARG.LS)
        LET S1.LS = -(BB.LS+SQ.LS)/(2.0*AA.LS)
        LET S2.LS =  (SQ.LS-BB.LS)/(2.0*AA.LS)
        IF S1.LS GE 1.0 CYCLE ELSE
        IF S2.LS LE MAX.BLK.LS CYCLE ELSE
        IF S2.LS LE 1.0
            LET SS.LS = S2.LS
            CALL LOS.TRE.BKG YIELDING BLOCKED
            IF BLOCKED EQ YES
                GO TO 'SAVE.ELL'     ''NO NEED TO CHECK AT S1
            OTHERWISE
        ALWAYS
        IF S1.LS GE 0.0
            LET SS.LS = S1.LS
            CALL LOS.TRE.BKG YIELDING BLOCKED
            IF BLOCKED EQ YES
                LET TRE.BLK.LS = S1.LS
                LET MAX.BLK.LS = S1.LS
            ALWAYS
        ALWAYS
        'SAVE.ELL'
        ADD 1 TO NELS.LS      LET IEL.LS(NELS.LS) = IC
        LET CS1.LS(NELS.LS) = S1.LS      LET CS2.LS(NELS.LS) = S2.LS
        IF CPK.LS GT CHTMAX.LS LET CHTMAX.LS = CPK.LS     ALWAYS
    LOOP  ''BACK FOR NEXT ELLIPSE IN THIS GRID SQUARE
    IF MAX.BLK.LS GT 0.01 ''NO REASON TO LOOK FURTHER SINCE O-T
        LET NGRSQ.LS = K  ''  EXTENSION IS ALREADY BLOCKED
        LEAVE   ''THE GRID SQUARE LOOP
    OTHERWISE
```

FIGURE II-10   (CONTINUED)

```
LOOP   ''BACK FOR NEXT GRID SQUARE
''   ALL ELLIPSES CHECKED AND SAVED
''  NOW START ON THE HILLS
'HILL.PROCESSING'
FOR K = 1 TO NGRSQ.LS DO
    LET IX.LS = IGX.LS(K)    LET IY.LS = IGY.LS(K)
    LET DUM.I(*) = LISTH(IX.LS,IY.LS,*)
    LET N = DIM.F(DUM.I(*))
    LET BASE.LS = DUM.I(1)
    FOR L = 2 TO N DO
        LET I = DUM.I(L)    '' GIVING THE HILL NUMBER
        IF KHREP(I) EQ KTREP CYCLE ELSE
        LET KHREP(I) = KTREP
'' COMPUTE W = TOP OF HILL I ALONG A TO B LINE
        LET PXX.LS = PXX.H(I)  LET PYY.LS = PYY.H(I)  LET PXY.LS = PXY.H(I)
        LET RX.LS = XA.LS - XC.H(I)     LET RY.LS = YA.LS - YC.H(I)
        LET GQ.LS = PXX.LS*XBASQ.LS + PYY.LS*YBASQ.LS + PXY.LS*XYBA.LS
        LET FQ.LS = 2.0*(PXX.LS*RX.LS*XBA.LS + PYY.LS*RY.LS*YBA.LS) +
            PXY.LS*(RX.LS*YBA.LS + RY.LS*XBA.LS)
        IF GQ.LS EQ 0.0 CYCLE ELSE
        LET W.LS = -FQ.LS / (2.0*GQ.LS)
        IF W.LS LE MAX.BLK.LS CYCLE ELSE
        IF W.LS GT 1.0 CYCLE ELSE
        IF ABS.F(W.LS) GT 5. CYCLE ELSE
        LET FSQ.LS = FQ.LS**2
        LET EQ.LS = PXX.LS*RX.LS**2 + PYY.LS*RY.LS**2 + PXY.LS*RX.LS*RY.LS
        LET POW.LS = EQ.LS - FSQ.LS / (4.0*GQ.LS)
        IF POW.LS LT -4.0 CYCLE ELSE
        LET PK.LS = PEAK.H(I)     LET HT.LS = HT.H(I)
        LET HHW.LS = PK.LS + HT.LS*(EXP.F(POW.LS)-1.)
        IF  HHW.LS LE BASE.LS CYCLE ELSE
        LET ZW.LS = ZA.LS + W.LS*ZBA.LS
        IF HHW.LS + CHTMAX.LS LT ZW.LS CYCLE ELSE
        IF NELS.LS EQ 0
            IF HHW.LS GT ZW.LS
                LET GRND.BLK.LS = W.LS
                LET MAX.BLK.LS = W.LS
                CYCLE
            OTHERWISE
        ELSE    ''CHECK WHETHER ANY FORESTS AT TOP OF HILL
            LET CVHTW.LS = 0.
            FOR M=1 TO NELS.LS WITH CS1.LS(M) LT W.LS AND CS2.LS(M) GT W.LS DO
                LET IC = IEL.LS(M)
                IF CVHTW.LS LT HT.E(IC)
                    LET CVHTW.LS = HT.E(IC)
                ALWAYS
```

FIGURE II-10 (CONTINUED)

```
                LOOP
                IF HHW.LS + CVHTW.LS LT ZW.LS
                    CYCLE
                OTHERWISE   ''A BLOCK - SEE IF GROUND OR TREES
                IF CVHTW.LS GT 0.1    '' IF ANY TREES, THEN TREES BLOCK
                    LET TRE.BLK.LS = W.LS
                ELSE        '' GROUND BLOCKS
                    LET GRND.BLK.LS = W.LS
                ALWAYS
                LET MAX.BLK.LS = W.LS
            ALWAYS
        LOOP   ''TO THE NEXT HILL FOR THIS GRIDSQUARE
    LOOP    ''TO THE NEXT GRID SQUARE
    '' HILLS DONE -- NOW FINALLY DETERMINE THE BACKGROUND CODE
    IF MAX.BLK.LS LT 0.01     ''NO BLOCK FOUND
        CALL ELEV(XA.LS, YA.LS) YIELDING ZZ.LS
        IF ZA.LS LT ZZ.LS           ''EXTENDED POINT UNDERGROUND
            IF MXBKGND LE 3
                LET BKGND = 1    ''UNSPECIFIED TERRAIN
            ELSE
                LET BKGND = 3    ''HILL SURFACE
            ALWAYS
            LET DISTOBKG = 0.9 * MAXDIST
            RETURN
        OTHERWISE
        LET DISTOBKG = MAXDIST
        IF ZBA.LS GT 0.0
            LET BKGND = 1           ''DISTANT TERRAIN
        ELSE
            LET BKGND = 2           '' SKY
        ALWAYS
    ELSE                       ''BLOCK FOUND
        LET DISTOBKG = MAXDIST * (1.0 - MAX.BLK.LS)
        IF MXBKGND LE 3
            LET BKGND = 1           ''UNSPECIFIED TERRAIN
        ELSE
            IF TRE.BLK.LS GT GRND.BLK.LS
                LET BKGND = 4    ''FOREST WITHIN MAXDIST
            ELSE
                LET BKGND = 3    ''GROUND WITHIN MAXDIST
            ALWAYS
        ALWAYS
    ALWAYS
    RETURN END
```

FIGURE II-10 (CONTINUED)

All three are initially set to zero. Values close to 1 indicate obstructions close to B. The closest of these values to 1 indicates the closest obstruction to B which then determines the background type.

Lines 47 - 79 Of the LOS.BKGND routine accumulate a list of the (Nominally 1 KM square) cross reference grid squares along the line from A to B. The sole purpose of these cross reference grids is to facilitate access to the hills and forest features which are close to the A/B line. (See Reference 3 for further details of the STAR terrain storage methods.)

Lines 80 - 87 Take care of the case where the extended O/T line is totally off the battlefield map (in which case detailed background computations are impossible).

Lines 89 - 140 Examine the forest features along the A/B line. For each such forest feature, if the A/B line intersects the forest ellipse, the S coordinates of the two intersection points are computed as S1 and S2 (with S1 < S2) in lines 107 and 108.

Since S1 and S2 mark the boundaries of a forest ellipse intersecting the A/B line, the routine checks whether the trees are high enough to obstruct the background line of sight by calling subroutine LOS.TRE.BKG with (global) input SS (= S1 or S2) and output BLOCKED (= YES or NO).

When SS = S2 we are at the forest boundary closest to B, and three situations can occur as illustrated in cross-section in Figure II-11. In situation (a) the A/B line lies below the ground at S2, so this test would update GRND.BLK.LS to S2. In situation (b) the A/B line intersects the forest boundary at S2, so TRE.BLK.LS is updated. Finally, in situation (c) the A/B line passes above the trees, so no obstruction occurs at S2. Note in Case (a) that S2 is not the actual location of the ground block (denoted SG on the Figure) but is only a rough lower bound. The routine does <u>not</u> solve for the

precise value of SG since this should require iterative solution of a transcendental equation, and the background type is the primary output from LOS.BKGND.

If S2 does not block the background LOS, then S1 is checked. Figure II-12 shows the two possible cases: In situation (d) the A/B line at S1 lies below the forest top, and thus a tree block occurs. TRE.BLK.LS is updated to S1 as a bound on ST. In situation (e) no blockage occurs at S1.

Lines 141 - 197 Examine the hilltops between A and B. The S coordinate of each hilltop is computed and denoted as W (Line 159).

If W is closer to B than any block so far, then the hilltop elevation HHW is computed and compared with the A/B line at W. Trees, if any, on top of the hill are also considered. The result may be either a ground or a tree block at W, or no block at all if the A/B line is high enough. Figure II-13 illustrates the 3 cases. In both situations f and g, where blocks occur, note that W is reported as a bound on the exact intersection locations SG and ST which are not computed.

Lines 198 - 228 Use the GRND.BLK.LS, TRE.BLK.LS, and MAX.BLK.LS Globals to sort out the proper background code and compute the DISTOBKG approximate distance.

Finally, the LOS.TRE.BKG routine which is called by LOS.BKGND is listed in Figure II-14. Its function should be clear from the above discussion.

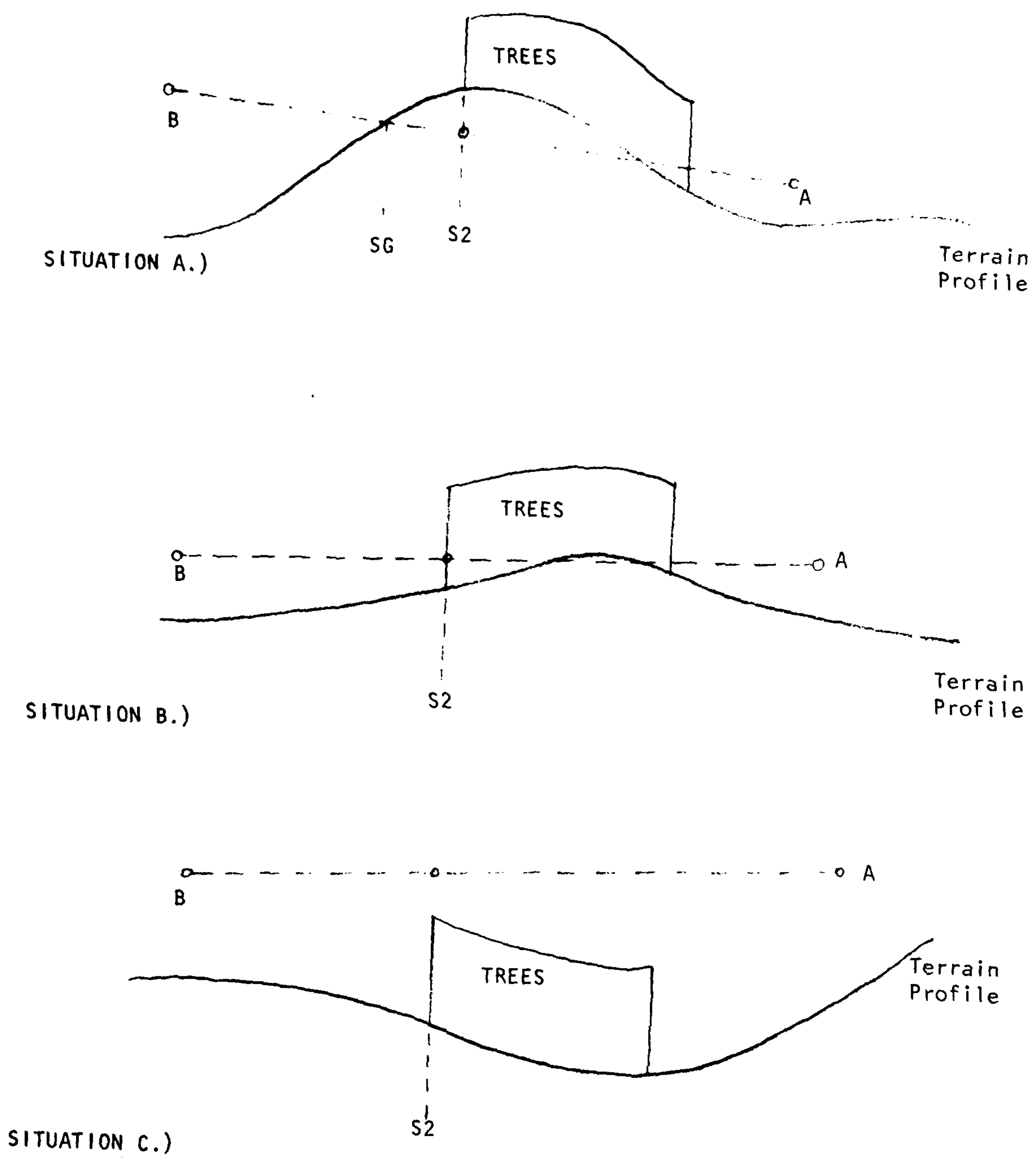
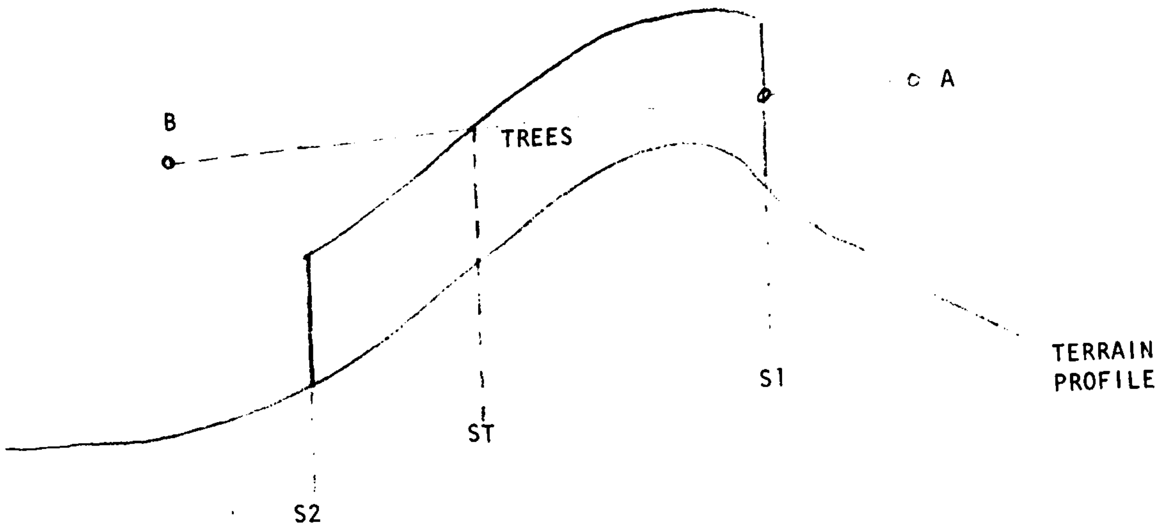
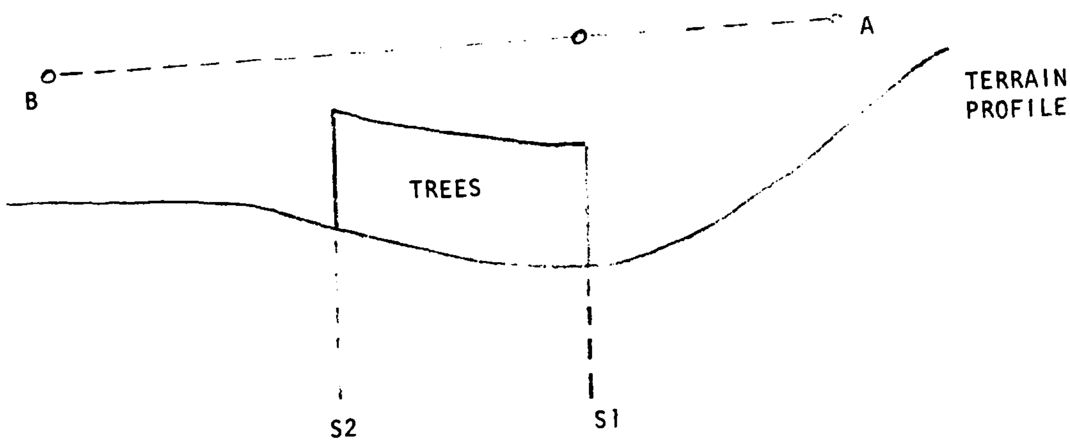


FIGURE II-11 THREE SITUATIONS WHEN SS=S2



SITUATION D.)



SITUATION E.)

FIGURE II-12 TWO CASES WHEN SS=S1

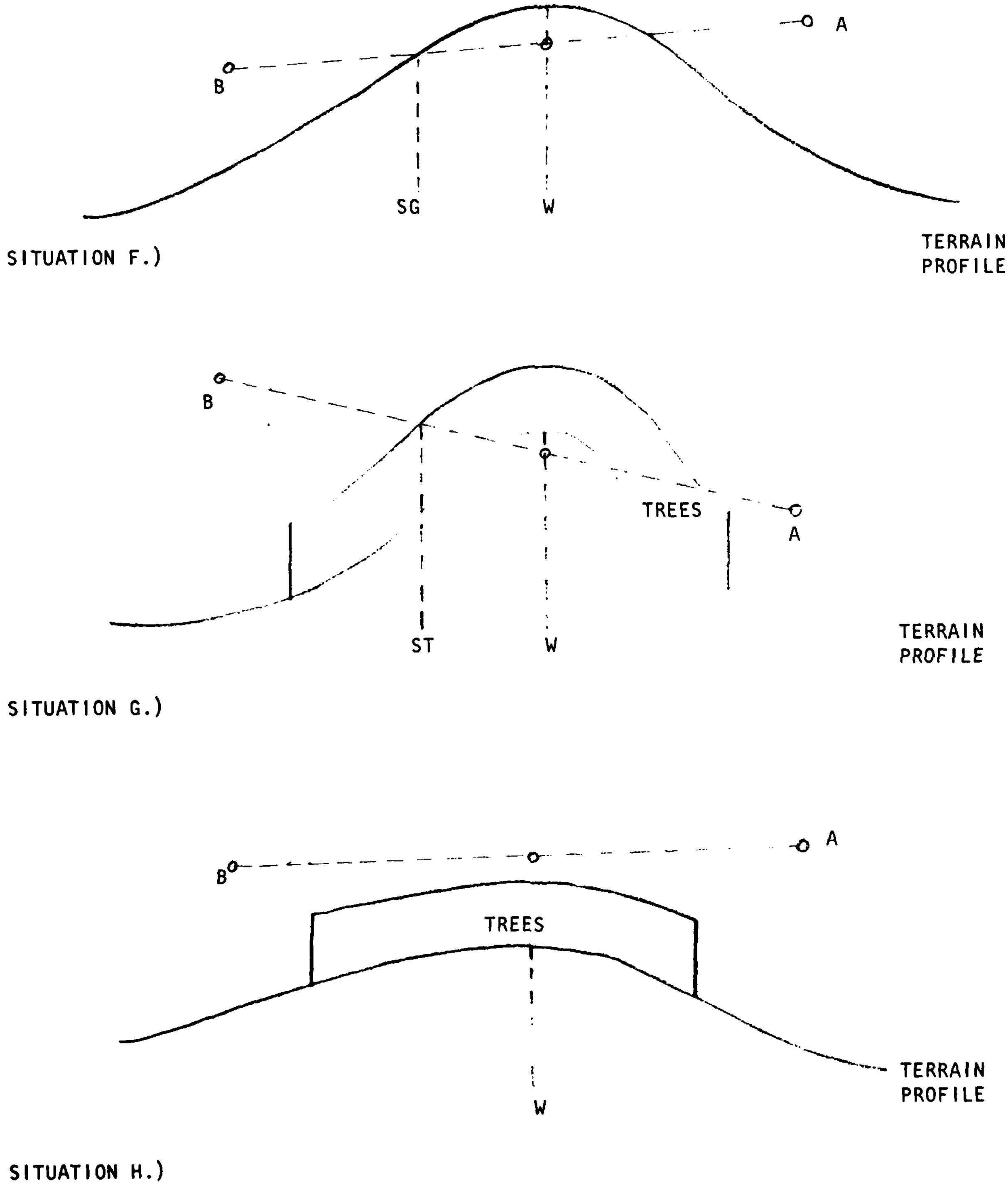


FIGURE II-13 THREE SITUATIONS AT HILL TOPS

```
ROUTINE LOS.TRE.BKG YIELDING BLOCKED
''************************************
DEFINE BLOCKED AS AN INTEGER VARIABLE
LET XS.LS = XA.LS + SS.LS*XBA.LS     LET YS.LS = YA.LS + SS.LS*YBA.LS
CALL ELEV GIVEN XS.LS, YS.LS YIELDING HTS.LS
LET ZS.LS = ZA.LS + SS.LS * ZBA.LS
IF ZS.LS GT HTS.LS + CPK.LS
    LET BLOCKED = NO
    RETURN
OTHERWISE
LET BLOCKED = YES
IF ZS.LS GT HTS.LS
    IF SS.LS GT TRE.BLK.LS
        LET TRE.BLK.LS = SS.LS
    ALWAYS
ELSE
    IF SS.LS GT GRND.BLK.LS
        LET GRND.BLK.LS = SS.LS
    ALWAYS
ALWAYS
IF SS.LS GT MAX.BLK.LS
    LET MAX.BLK.LS = SS.LS
ALWAYS
RETURN
END
```


FIGURE II-14 ROUTINE LOS.TRE.BKG

III. DEFINING AND MANIPULATING THE DETECTED LIST

A. THE DETECTED LIST

The NVL target acquisition methodology defines several levels of acquisition corresponding to increasing resolution in the target image. Since the DYNTACS detection model, which was originally used in STAR, had only one acquisition level (identification) substantial changes to the detected list structure in STAR are required. These changes will affect many of the current STAR routines and events. In this Chapter we discuss the new detected list structure, the new routines for manipulating the list, and the changes required in the current STAR code.

Each combat entity, A, in the STAR model has a list of enemy entities whose locations are known to A. This list is called the detected list and is denoted in the STAR code by the name LIST. New detection events add to the list, while deaths or loss of intervisibility remove targets from the list. The list must be accessed whenever a target selection event occurs. Additions, removals, and target selections happen relatively rarely in execution of the STAR model. A far more frequent occurrence, which happens every time entity A attempts to detect any potential target, is a check to see if that target is already on the list.

Since list searches occur far more frequently than list additions or deletions, computational efficiency argues for a LIST data structure which allows random access (rather than sequential access) so that a binary search may be employed. Thus we implement LIST as a simple 2 x M array for each combat entity where M is either the number of targets currently on the list, or 1 if no targets are on the list. The array data structure allows efficient

searching, but requires that the entire array be redefined whenever its size changes. (The alternative dynamic set or linked list data structures would make additions and deletions easier but require a linear search.)

The global name LIST is used in common for accessing every entity's detected list. As the LIST for element A is reserved, its pointer, LIST (*,*), is saved in the array TRGT (1, NAME(A)) for future reference to A's list. To access the detected list for any element A, then, we simply

LET LIST (*,*) = TRGT (1, NAME(A))

As a convention in the model, any routine which has thus referred to a LIST should perform

LET LIST (*,*) = 0

prior to RETURN. This has made early detection of some programming errors substantially simpler.

In what follows, we assume that the pointer LIST (*,*) has been set to the detected list for the element whose entity pointer is A. If A's detected list has no targets on it, then LIST (1,1) = 0 and LIST (2,1) is ignored (and the LIST has dimension 2 x 1). If the detected list contains M target ($M \geq 1$), then for J = 1, ..., M we define

LIST (1,J) = entity pointer of the Jth target on the list

LIST (2,J) = acquisition level code for Jth target

The STAR model will frequently check A's detected list to see whether element B is on the list. To speed this check we require that the target pointers in LIST (1,J) are stored in increasing numerical order.

In the process of adding the second dimension to the target list, several existing STAR routines were modified. In particular, the old LIST.UPDATE routine was separated into two new routines LIS.ADD and LIS.DELETE

to eliminate the confusing WHO CALLED argument. Also TARGET.SELECT scheduling was removed from the LIST manipulating routines and put in the search tactics routines instead. The rest of this Chapter details the new list routines and modifications made to other existing STAR events and routines.

B. ROUTINE LIS.ADD

Purpose: The LIS.ADD routine adds an element B to A's detected list in proper sequence (if B is not there already). The acquisition level of B may be increased if B is already on A's list at a lower level.

GIVEN ARGUMENTS

A	INTEGER	Pointer to observer
B	INTEGER	Pointer to target
ACQ.LEV	INTEGER	Level at which A has acquired B

YIELDING ARGUMENT

SIZE	REAL	Number of elements on the list on return

LOCAL VARIABLES AND ARRAYS

ANSWER	INTEGER	Result of call to LIS .CHECK. YES (= 1) if B is already on A's List, no otherwise
DIM	INTEGER	Size of A's list on entry
I	INTEGER	Loop Index
OLD.LEV	INTEGER	Result of call to LIS.CHECK - If B is on A's list OLD.LEV = current acquisition level code
POS	INTEGER	Result of call to LIS.CHECK. If B is on A's list already, POS indicates B's position on the list. If B is not on A's list, then POS is the position of the target after which B should be inserted.
TEMP	INTEGER 2-D	Temporary array for holding list's contents while list is reserved one larger.

GLOBAL ARRAYS

LIST	INTEGER 2-D	A's detected list
TRGT	INTEGER 2-D	Storage for pointer to A's detected list

ENTITY ATTRIBUTES

NAME	INTEGER	ID number of entity A

ROUTINES CALLED

DIM.F	Gives array size
LIS.CHECK	Checks to see if B is already on A's list.

EVENT SCHEDULED

None

SIMSCRIPT CODE

See Figure III-1.

LINE BY LINE COMMENTARY (LIS.ADD)

Lines 1 - 7	Declare the routine and define variables.
Line 8	Tests whether B is already on the list.
Line 9	Accesses A's current list calling it TEMP.
Lines 10 - 15	Handle the case where B is already on A's list so at most a change in acquisition level is required.
Lines 16 - 22	Handle the case where A's list is empty, so B becomes the sole entry on the list.
Lines 23 - 28	Create a new list which is one larger than TEMP.
Lines 29 - 32	Transfer the front part of TEMP to list.
Lines 33 - 34	Insert B in the proper position in list.
Lines 35 - 38	Transfer the remainder of TEMP to list.
Line 39	Releases the old detected list for A since a new one has been created.
Lines 40 - 42	Terminate the routine

```
ROUTINE LIS.ADD(A,B,ACQ.LEV) YIELDING SIZE
''*********************************************
''ADDS B TO THE DETECTED LIST OF A OR INCREASES ACQUISITION LEVEL IF B IS
''ALREADY ON LIST AT LOWER LEVEL
DEFINE A,B,ANSWER,ACQ.LEV,POS,OLD.LEV,DIM,I AS INTEGER VARIABLES
DEFINE SIZE AS A REAL VARIABLE
DEFINE TEMP AS A 2-DIMENSIONAL INTEGER ARRAY
CALL LIS.CHECK(A,B) YIELDING ANSWER,OLD.LEV,POS,SIZE
LET TEMP(*,*) = TRGT(1,NAME(A))   ''DETECTED LIST FOR A
IF ANSWER EQ YES    ''B ALREADY ON LIST
   IF ACQ.LEV GT OLD.LEV
      LET TEMP(2,POS) = ACQ.LEV
   ALWAYS
   LET TEMP(*,*) = 0
   RETURN
OTHERWISE        ''ANSWER IS NO, SO B NOT ON LIST
IF TEMP(1,1) = 0   ''LIST EMPTY
   LET TEMP(1,1) = B
   LET TEMP(2,1) = ACQ.LEV
   LET SIZE = 1.0
   LET TEMP(*,*) = 0
   RETURN
OTHERWISE     ''LIST MUST BE MADE LARGER TO ADD B
LET DIM = DIM.F(TEMP(1,*))
LET LIST(*,*) = 0
RESERVE LIST(*,*) AS 2 BY DIM+1
LET SIZE = DIM + 1
LET TRGT(1,NAME(A)) = LIST(*,*)
FOR I = 1 TO POS DO
   LET LIST(1,I) = TEMP(1,I)
   LET LIST(2,I) = TEMP(2,I)
LOOP
LET LIST(1,POS+1) = B
LET LIST(2,POS+1) = ACQ.LEV
FOR I = POS+1 TO DIM DO
   LET LIST(1,I+1) = TEMP(1,I)
   LET LIST(2,I+1) = TEMP(2,I)
LOOP
RELEASE TEMP(*,*)
LET LIST(*,*) = 0
RETURN
END
```

FIGURE III-1   ROUTINE LIS.ADD

C. ROUTINE LIS.CHECK

Purpose: Routine LIS.CHECK performs a binary search to determine whether element B is on A's detected list. If so, it returns position in the list and the current acquisition level from the list. If B is not on A's list, the routine returns the position of the target after which B should be inserted.

GIVEN ARGUMENTS

A	INTEGER	Pointer to Observer
B	INTEGER	Pointer to Target

YIELDING ARGUMENTS

ANSWER	INTEGER	Result - YES (= 1) if B is on the list NO (= 0) if B is not on list
ACQ.LEV	INTEGER	Current acquisition level from LIST if Answer = Yes (Otherwise 0).
POS	INTEGER	If Answer = YES, B's position on the list. Otherwise, position of target after which B should be added.
SIZE	REAL	Number of targets on A's detected list.

LOCAL VARIABLES

LO	INTEGER	A test position in list to left of B
HI	INTEGER	A test position in list to right of B
MID	INTEGER	TEST Position - Midway between LO and HI
MIDPOINTER	INTEGER	Entity pointer stored at position MID in LIST

GLOBAL ARRAYS

LIST	INTEGER 2-D	A's detection list
TRGT	INTEGER 2-D	Storage for pointer to A's list

ENTITY ATTRIBUTES

NAME INTEGER ID number of entity A

ROUTINE CALLED

DIM.F Gives array size

EVENTS SCHEDULED

None

SIMSCRIPT CODE

See Figure III-2.

LINE BY LINE COMMENTARY (LIS.CHECK)

Lines 1 - 8	Declare the routine and define variables.
Line 9	Accesses A's detected list.
Lines 10 - 14	Handle the case where no search is required because A's list is empty.
Lines 15 - 17	Set up initial conditions for the search.
Line 18	Is the search failure condition tested at the end of each pass through the loop.
Lines 19 - 20	Place the test value at the midpoint of the remaining interval of uncertainty.
Lines 21 - 25	Reduce the interval of uncertainty if no match is found.
Lines 26 - 30	Terminate the search if a match is found.
Lines 34 - 36	Terminate the search if no match is found.
Lines 37 - 40	Terminate the routine.

```
ROUTINE LIS.CHECK(A,B) YIELDING ANSWER,ACQ.LEV,POS,SIZE
''*******************************************************
''CHECKS TO SEE IF B IS ON DETECTED LIST OF A.  IF SO, RETURNS ACQUISITION
''LEVEL, POSITION IN LIST, AND LIST SIZE.  IF NOT, RETURNS POSITION IN
''LIST AFTER WHICH B SHOULD BE INSERTED.

DEFINE A,B,ANSWER,ACQ.LEV,POS,LOW,HI,MID,MIDPOINTER AS INTEGER VARIABLES
DEFINE SIZE AS A REAL VARIABLE
LET LIST(*,*) = TRGT(1,NAME(A))
IF LIST(1,1) EQ 0  ''LIST IS EMPTY
   LET SIZE = 0.0
   LET POS = 0
   LET ANSWER = NO
ELSE
   LET LOW = 1
   LET HI = DIM.F(LIST(1,*))
   LET SIZE = HI
   UNTIL LOW GT HI DO
      LET MID = TRUNC.F((LOW+HI+0.1)/2)
      LET MIDPOINTER = LIST(1,MID)
      IF B GT MIDPOINTER
         LET LOW = MID + 1
      ELSE
         IF B LT MIDPOINTER
            LET HI = MID - 1
         ELSE  ''B = MIDPOINTER AND SEARCH ENDS WITH A MATCH
            LET POS = MID
            LET ANSWER = YES
            LET ACQ.LEV = LIST(2,MID)
            GO TO 'OUT'
         ALWAYS
      ALWAYS
   LOOP
''LOW GT HI SO SEARCH HAS FAILED
   LET ANSWER = NO
   LET POS = HI   ''POSITION AFTER WHICH TO INSERT B
ALWAYS
'OUT'  LET LIST(*,*) = 0
RETURN
END
```

FIGURE III-2   ROUTINE LIS.CHECK

D. ROUTINE LIS.DELETE

Purpose: Removes B from A's detection list if B is on the list.

GIVEN ARGUMENTS

A	INTEGER	Pointer to observer
B	INTEGER	Pointer to Target

YIELDING ARGUMENT

SIZE	REAL	Number of elements on the list on return

LOCAL VARIABLES AND ARRAYS

ANSWER	INTEGER	Result of call to LIS.CHECK
ACQ.LEV	INTEGER	
POS	INTEGER	
DIM	INTEGER	Size of A's list on entry
I	INTEGER	Loop counter
TEMP	INTEGER 2-D	Array for holding list's contents while list is reserved one smaller

GLOBAL ARRAYS

LIST	INTEGER 2-D	A's detected list
TRGT	INTEGER 2-D	Storage for pointer to A's detected list

ENTITY ATTRIBUTES

NAME	INTEGER	ID number of entity A

ROUTINE CALLED

DIM.F	Gives array size
LIS.CHECK	To see if B is on A's list

EVENTS SCHEDULED

None

SIMSCRIPT CODE

See Figure III-3

LINE BY LINE COMMENTARY (LIS.DELETE)

Lines 1 - 6	Declare the routine and define variables.
Line 7	Locates B in the List (if it is there).
Lines 8 - 12	Access A's detected list and call it TEMP.
Lines 13 - 15	Handle the case where B is the only element on A's list.
Lines 16 - 17	Reserve a new smaller list for the case where B is not the only target on the list.
Lines 18 - 21	Copy list entries before B from TEMP to list.
Lines 22 - 25	Copy list entries after B from TEMP to list.
Line 27	Saves the new list pointer.
Lines 28 - 32	Terminate the routine.

```
ROUTINE LIS.DELETE(A,B) YIELDING SIZE
''*************************************
''REMOVES B FROM A'S DETECTED LIST (IF B IS ON IT) AND RETURNS LIST SIZE
DEFINE A,B,ANSWER,ACQ.LEV,POS,DIM,I AS INTEGER VARIABLES
DEFINE SIZE AS A REAL VARIABLE
DEFINE TEMP AS A 2-DIMENSIONAL INTEGER ARRAY
CALL LIS.CHECK(A,B) YIELDING ANSWER, ACQ.LEV,POS,SIZE
IF ANSWER EQ YES
   LET TEMP(*,*) = TRGT(1,NAME(A))
   LET DIM = DIM.F(TEMP(1,*))
   LET SIZE = DIM - 1.0
   LET LIST(*,*) = 0
   IF DIM EQ 1
      RESERVE LIST(*,*) AS 2 BY 1
      LET LIST(1,1) = 0
   ELSE
      RESERVE LIST(*,*) AS 2 BY DIM-1
      FOR I = 1 TO POS-1 DO
         LET LIST(1,I) = TEMP(1,I)
         LET LIST(2,I) = TEMP(2,I)
      LOOP
      FOR I = POS TO DIM-1 DO
         LET LIST(1,I) = TEMP(1,I+1)
         LET LIST(2,I) = TEMP(2,I+1)
      LOOP
   ALWAYS
   LET TRGT(1,NAME(A)) = LIST(*,*)
   RELEASE TEMP(*,*)
ALWAYS
LET LIST(*,*) = 0
RETURN
END
```

FIGURE III-3   ROUTINE LIS.DELETE

E. ROUTINE LIS.PURGE

Purpose: Removes elements from A's list if they are dead or no longer visible. In the process, updates location of A and elements on A's list. Typically called from TARGET.SELECT prior to selection. This routine is a rewrite of the old PURGE.LIST routine.

GIVEN ARGUMENT

A	INTEGER	Pointer to observer

LOCAL VARIABLES AND ARRAYS

B	INTEGER	Pointer to elements on A's list
DIM	INTEGER	Size of A's list on entry
I	INTEGER	Loop counter
SIZE	REAL	Result of LIS.DELETE call - not used
CHECKER	INTEGER 1-D	Temporary array to hold pointers from A's list while checking visibility

GLOBAL VARIABLES AND ARRAY

FWD.LOOK	INTEGER	Set direction of LOS call for the sight routine.
BWD.LOOK	INTEGER	
CRITIC.VALUE	REAL	LOS threshold.
PCT.VIS	REAL	Returned percent visible from sight call
LIST	INTEGER 2-D	A's detected list
TRGT	INTEGER 2-D	Storage for pointer to A's detected list

ENTITY ATTRIBUTES

ALIVE.DEAD	INTEGER	1 = DEAD, 0 = ALIVE
DEFNUM	INTEGER	Deflilade condition
NAME	INTEGER	ID number of entity

ROUTINES CALLED

DIM.F	Gives array size.
LIS.DELETE	Removes element from A's list.
LOC	Updates location of an element.
SIGHT	Checks line of sight between two elements.

EVENTS SCHEDULED

None

SIMSCRIPT CODE

See Figure III-4.

LINE BY LINE COMMENTARY (LIS.PURGE)

Lines 1 - 7	Declare the routine and define variables.
Line 9	Updates A's position.
Line 10	Accesses A's detected list.
Lines 11 - 14	If list is empty, return.
Lines 15 - 17	Copy target pointers from list into the temporary array checker.
Lines 18 - 20	Set up for call to sight.
Lines 21 - 22	Loop over each B on A's list.
Lines 23 - 24	Check if B is dead or in full defilade and if so remove B from the list.
Line 26	Updates B's location.
Lines 27 - 28	Check if LOS exists from A to B.
Lines 30	Removes B from A's list.
Lines 33 - 35	Release temporary storage and terminate the routine.

```
ROUTINE LIS.PURGE(A)
''******************
'' REMOVES ELEMENTS FROM A'S DETECTED LIST IF THEY ARE DEAD OR NO LONGER
''      VISIBLE.  IN THE PROCESS, UPDATES POSITION OF A AND ALL ELEMENTS
''      REMAINING ON THE LIST.
DEFINE A, B, DIM, I AS INTEGER VARIABLES
DEFINE SIZE AS A REAL VARIABLE
DEFINE CHECKER AS A 1-DIMENSIONAL INTEGER ARRAY
CALL LOC(A)
LET LIST(*,*) = TRGT(1,NAME(A))
IF LIST(1,1) = 0
   LET LIST(*,*) = 0
   RETURN
OTHERWISE
LET DIM = DIM.F(LIST(1,*))
RESERVE CHECKER(*) AS DIM
FOR I = 1 TO DIM LET CHECKER(I) = LIST(1,I)
LET LIST(*,*) = 0
LET FWD.LOOK = 1
LET BWD.LOOK = 0
FOR I = 1 TO DIM DO
   LET B = CHECKER(I)
   IF ALIVE.DEAD(B) EQ 1 OR DEFNUM(B) EQ 1
      GO TO REMOVE
   OTHERWISE
   CALL LOC(B)
   CALL SIGHT(A,B)
   IF PCT.VIS LT CRITICAL.VALUE
      'REMOVE'
      CALL LIS.DELETE(A,B) YIELDING SIZE
   ALWAYS
LOOP
RELEASE CHECKER
RETURN
END
```

FIGURE III-4   ROUTINE LIS.PURGE

F. ROUTINE LIS.RELEASE

Purpose: The LIS.RELEASE routine totally erases A's detected list. It is used when A goes to a full defilade condition.

GIVEN ARGUMENT

A	INTEGER	Pointer to entity

GLOBAL ARRAYS

LIST	INTEGER 2-D	A's detected list
TRGT	INTEGER 2-D	Storage for pointer to A's list

ENTITY ATTRIBUTE

Name	INTEGER	A's ID number

SIMSCRIPT CODE

See Figure III-5.

COMMENTARY

The routine is self-explanatory.

```
ROUTINE LIS.RELEASE(A)
''**********************
'' TOTALLY ERASES THE DETECTED LIST FOR ELEMENT A
DEFINE A AS AN INTEGER VARIABLE
LET LIST(*,*) = TRGT(1,NAME(A))
RELEASE LIST(*,*)
RESERVE LIST(*,*) AS 2 BY 1
LET TRGT(1,NAME(A)) = LIST(*,*)
LET LIST(1,1) = 0
LET LIST(*,*) = 0
RETURN
END
```

FIGURE III-5   ROUTINE LIS.RELEASE

G. ROUTINE LIS.LEVEL.PURGE

Purpose: The LIS.LEVEL.PURGE routine removes entities from A's detected list if their acquisition level is lower than a specified value.

GIVEN ARGUMENT

A	INTEGER	Pointer to Entity
LVL	INTEGER	Acquisition level threshold for removal

LOCAL VARIABLES

B	INTEGER	Pointer to entity on A's list
I	INTEGER	Loop index
DIM	INTEGER	Size of A's list
J	INTEGER	Loop index
SIZE	REAL	Returned from LIS.DELETE - not used
TEMP	INTEGER 2-D	Temporary storage for A's list

GLOBAL ARRAYS

LIST	INTEGER 2-D	A's detected list
TRGT	INTEGER 2-D	Storage for pointer to A's list

ENTITY ATTRIBUTES

NAME	INTEGER	A's ID number

ROUTINES CALLED

DIM.F	To get size of A's list
LIS.DELETE	To remove B from A's list

EVENTS SCHEDULED

None

SIMSCRIPT CODE

See Figure III-6.

LINE BY LINE COMMENTARY

Lines 1 - 6	Declare the routine and define variables.
Line 7	Accesses A's detected list.
Lines 8 - 11	Take care of an empty list.
Lines 12 - 17	Transfer A's list to the TEMP array.
Lines 18 - 23	Loop through the list, possibly deleting entities from it.
Lines 24 - 26	Release temporary storage and terminate the routine.

```
ROUTINE LIS.LEVEL.PURGE(A,LVL)
''********************************
''REMOVES ELEMENTS FROM A'S DETECTED LIST IF THEIR ACQUISITION LEVEL IS LT LVL
DEFINE A, LVL, B, I, DIM, J AS INTEGER VARIABLES
DEFINE SIZE AS A REAL VARIABLE
DEFINE TEMP AS A 2-DIMENSIONAL INTEGER ARRAY
LET LIST(*,*) = TRGT(1,NAME(A))
IF LIST(1,1) EQ 0
    LET LIST(*,*) = 0
    RETURN
OTHERWISE
LET DIM = DIM.F(LIST(1,*))
RESERVE TEMP(*,*) AS 2 BY DIM
FOR I = 1 TO DIM
FOR J = 1 TO 2
    LET TEMP(J,I) = LIST(J,I)
LET LIST(*,*) = 0
FOR I = 1 TO DIM DO
    IF TEMP(2,I) LT LVL
        LET B = TEMP(1,I)
        CALL LIS.DELETE(A,B) YIELDING SIZE
    ALWAYS
LOOP
RELEASE TEMP(*,*)
RETURN
END
```

FIGURE III-6   ROUTINE LIS.LEVEL.PURGE

H. INCORPORATING THE 2-D LIST INTO STAR

In the course of building the 2-Dimensional list structure, in preparation for the new target acquisition module for STAR, many routines and events of the existing STAR model had to be modified. This section names the program segments affected with a brief description of the nature of the changes. The precise lines to be changed depend on the STAR version being updated.

1. Preamble

   a. EACH TANK (OR UNIT) has a single new attribute SCH.TYPE.

   b. RES.SCH (new routine) is declared releaseable.

   c. EVENT NOTICES INCLUDE:

   SITUATION.UPDATE (this event replaces STEP.TIME. It updates positions and does movement coordination checks, but no detection computations). DETECT (number of arguments changed) SEARCH (new event).

   d. GLOBAL VARIABLES

LOC.DELTA.T	(REAL)	Frequency of SITUATION.UPDATE scheduling
TEMP.TGT		DELETE
LIST		Changed to 2-Dimensional (vice 1-Dimensional)
SCH.DATA	(REAL,3-D)	For data defining the search types
TYPE.SCH	(INTEGER, 2-D)	Search types for each system/weapon type)

(NOTE NVL data arrays must also be added as in Chapter II Section D)

2. Main

   a. Call RES.SCH
   Release RES.SCH — To initialize data for search module

b. Read LOC.DELTA.T

c. Schedule SITUATION.UPDATE

3. BL.CREATE

a. Set SCH.TYPE for each TANK.

b. Schedule SEARCH for each observer on TANK.

c. Reserve and initialize 2-Dimensional detected LIST.

4. DETECT

a. Add ACQ.LEV (acquisition level) as a given argument.

b. Event has been rewritten using new list routines.

c. Target selection now scheduled in DETECT vice in LIST.UPDATE.

d. Negative pointer for flash detection no longer used.

e. Addition of B to list now done here rather than in PROXIMITY.DETECT.

5. FIRE

Flash stimulus detection changed, replacing negative pointer with ACQ.LEV of 1.

6. IMPACT

Removal from list now handled by call to LIS.DELETE.

7. LIST.UPDATE

Replaced by LIS.ADD and LIS.DELETE. Note that these no longer schedule target selection.

8. SITUATION.UPDATE

a. New event to periodically update position for every element on the battlefield.

b. Reschedules itself in LOC.DELTA.T units.

9. PROXIMITY.DETECT

Total rewrite - now only adds elements close to B to A's list. B itself is added in DETECT.

10. PURGE.LIST

Replaced by LIS.PURGE - total rewrite to handle 2-D list structure but essentially the same function.

11. RED.CREATE

a. Set SCH.TYPE for each TANK.

b. Schedule SEARCH for each observer on TANK

c. REserve and initialize 2-D detected LIST.

12. RES.SCH

New releaseable routine for reserving and reading all data for target acquisition module.

13. SEARCH

a. New Event - looks up an observer's search tactic and calls appropriate STKn routine.

b. Reschedules self in time used by one search cycle. (See Chapter V.)

14. STK1, STK2 etc.

Search tactic routines - See Chapter VI.

15. STEP.TIME - Deleted

16. TACTICS

References to LIST are adjusted to account for new 2-Dimensional LIST structure.

17. TALLY.HIT.STATE

    2-D LIST Changes.

18. TARGET.SELECT

    a. Call LIS.PURGE vice PURGE.LIST

    b. 2-D LIST Change.

IV. ASSOCIATING SENSORS AND SEARCH TACTICS WITH SIMULATED COMBATANTS

A. THE SCH.TYPE ATTRIBUTE

The STAR Target Acquisition Module is designed to allow an arbitrary number of observers for each combat entity. For example, an entity which simulates a tank might have two observers corresponding to the tank commander and the gunner. Each observer may have several sensors which are used in various ways and circumstances. A design goal for the Target Acquisition Module has been to model not only the physical behavior of the sensors, but also to present a versatile and flexible structure within which a wide variety of search tactics and sensor device useage patterns may be investigated.

Observers, sensors, and search tactics are associated with STAR combat entities using a single attribute defining the "search type" for each entity. This integer attribute, the SCH.TYPE is an index into a global 3-Dimensional real array, SCH.DATA, which details the observers, sensors, and search tactics for that combat entity.

There is no model imposed limit on the number of SCH.TYPE's which may be created. At one extreme all entities in the simulation might use the same SCH.TYPE. This would imply that they all had the same number of observers, the same sensors, and the same procedures for choosing and using the sensors. At the other extreme, the model user might define a separate SCH.TYPE for each individual combat entity. A middle ground which will often be useful is to let the SCH.TYPE be a function of the system type/weapon type categories being simulated.

As the model is currently configured, the user must input SCH.TYPE value for each system type/weapon type category (however the SCH.TYPE input may be the same for several different categories). A simple code change would allow the option of overriding this SCH.TYPE assignment for any particular entities as desired. (For example the tank of an armor company commander might be equipped with a sensor configuration different from that of other tanks in the company. It would then need a distinct SCH.TYPE attribute.)

B. THE SCH.DATA ARRAY

The user defines the meaning of each SCH.TYPE by entering data for the SCH.DATA array. This 3-dimensional real ragged array has subscripts:

TYPE - The search type (ranging from 1 up to MXTYPE, the Maximum Type used in this run)

OBS - Observer number (ranging from 1 to NOBS, the number of observers for this search type)

J - Index for parameters to define the sensors and tactics for this observer.
- J = 1 code for the search tactic to be used by this observer.
- J = 2,...N Parameters which further define the tactic (such as the sensor to use). The number of these parameters and their meaning depends on the search tactic being used, and will be discussed at length in the Chapters devoted to individual search tactics.

Each SCH.TYPE is thus associated with:

1. A number of observers for the combat entity.
2. For each observer a search tactic code, and
3. Parameters to further define the tactic, such as sensor(s) to use, time to spend searching, acquisition level to strive for, etc.

The actual implementation of the search is done by the SEARCH event in conjunction with several search tactics routines. These computer programs will be discussed in Chapters V and VI.

Data for the SCH.DATA array is input in routine RES.SCH which was discussed in Chapter II.

V. THE SEARCH EVENT

A. DISCUSSION

Target acquisition computations in STAR are driven by a SEARCH event which is scheduled to occur periodically for each searching observer on the battlefield. When the SEARCH event for a given observer occurs, the SEARCH event looks up the observer's search tactic and sensor equipment in the SCH.DATA array and then calls the appropriate search tactics routine to actually do the acquisition computations. If the computations indicate that target acquisitions should occur, then DETECT events are scheduled. The amount of time, T, used by one search cycle is computed by the search tactics routine and returned to the SEARCH event. This time may depend on whether the search was successful. The SEARCH event will then reschedule itself to resume searching after T time units have passed.

B. PROGRAM DOCUMENTATION - SEARCH

Purpose. The SEARCH event coordinates target acquisition computations for each observer by calling an appropriate search tactics routine.

GIVEN ARGUMENTS

A	INTEGER	Pointer to searching entity
OBS	INTEGER	Observer number on that entity
TYPE	INTEGER	The search type to use for this occurrance of the search event.

LOCAL VARIABLES

TAC	INTEGER	Search tactic to be used by the observer
TIME.USED	REAL	Amount of time used by search tactics routine
NEWTYPE	INTEGER	Search type to use for the next occurrance of the search event for this observer

GLOBAL VARIABLES

SCH.DATA	REAL 3-D	Definition of A's search type.

ENTITY ATTRIBUTES

ALIVE.DEAD	INTEGER	A's status.

ROUTINES CALLED

STKn	Search tactics routine for tactic n

EVENTS SCHEDULED

SEARCH	Recursively schedule next search for this observer

SIMSCRIPT CODE

See Figure V-1

COMMENTARY

The SEARCH event is largely self-explanatory. Note, however, the use of the subscripted labels in Line 11. Care should be taken when adding new search tactics that the upper bound on TAC in Line 10 is changed to reflect the new routine.

As written here, SEARCH can call several search tactics routines. These will be documented in the next Chapter.

```
EVENT SEARCH(A,OBS,TYPE)
''**************************************************************************
'' SEARCH EVENT FOR OBSERVER OBS OF ENTITY A USING GIVEN SEARCH TYPE
DEFINE A, OBS, TYPE, NEWTYPE, TAC AS INTEGER VARIABLES
DEFINE TIME.USED AS A REAL VARIABLE
IF ALIVE.DEAD(A) NE 0
   RETURN
OTHERWISE
LET TAC = SCH.DATA(TYPE,OBS,1)
IF TAC GE 1 AND TAC LE 4  '' CHANGE UPPER LIMIT WHENEVER A NEW TACTIC IS ADDED
   GO TO 'SCHTAC(TAC)'
OTHERWISE
PRINT 1 LINE WITH NAME(A), OBS, TYPE, TAC, TIME.V AS FOLLOWS
XXX ERROR IN SEARCH - NAME ***** OBS *** SCH.TYPE *** S TACTIC **** TIME ****.**
RETURN
'SCHTAC(1)'
   CALL STK1(A,OBS,TYPE) YIELDING TIME.USED, NEWTYPE  ''DYNTACS VISUAL
   GO TO 'RESCHED'
'SCHTAC(2)'
   CALL STK2(A,OBS,TYPE) YIELDING TIME.USED, NEWTYPE  ''NVL SINGLE DEVICE
   GO TO 'RESCHED'
'SCHTAC(3)'
   CALL STK3(A,OBS,TYPE) YIELDING TIME.USED, NEWTYPE  ''AIR/ADA VISUAL
   GO TO 'RESCHED'
'SCHTAC(4)'
   CALL STK4(A,OBS,TYPE) YIELDING TIME.USED, NEWTYPE  ''ADA RADAR
   GO TO 'RESCHED'
'RESCHED'
SCHEDULE A SEARCH(A,OBS,NEWTYPE) IN TIME.USED UNITS
RETURN
END
```

FIGURE V-1   EVENT SEARCH

VI. SEARCH TACTICS - ROUTINES

A. THE CONCEPT OF A SEARCH TACTIC

The incorporation of the NVL search model into the STAR combat simulation makes it possible to simulate a wide variety of target acqusition devices and situations. This capability to simulate multiple observers, multiple sensors, various modes of sensor use and various levels of target acquisition creates an obligation for the model builder and user to coooperate in defining realistic modes of employment for the sensors that are made available to each observer. These modes of sensor employment will be called search tactics.

The search tactic for a given observer will typically include the following sorts of computations:

1. Preliminary Target List Managment. If the observer has moved into a full defilade position, his entire target list might be erased or the acquisition level might be lowered for targets on the list, thus requiring some effort for reacquisition when he emerges from defilade. Transient target signatures such as gun flashes which have not been upgraded to higher acquisition levels during the previous search event might be removed from the list.

2. Determine Area to Search. Each entity in the simulation has a primary direction of search related to its sector of responsibility. The search tactic must decide whether to search the entire sector during this search cycle, or whether to concentrate on some smaller subarea possibly around a direction in which searches have recently been successful. Alternatively the tactic may decide not to search, but rather to "stare" at already localized targets in an attempt to upgrade their acquisition levels.

3. <u>Create a Set of Potential Targets</u>. Usually, only a small subset of the elements on the battlefield are in a position to be detected by a particular observer. Simple tests may be used to screen out obviously ineligible targets. Examples include enemy/friendly tests, range checks, sector checks, mounted/dismounted checks, and line of sight tests. Targets which pass the screening tests can be filed in the potential target set in order of (for example) range so that closer targets will be considered first in the detection computations. Also, some systems, such as air defense, are only interested in particular enemy elements (e.g. air) so only those would be filed in the set.

4. <u>Specify Sensor Device Utilitation</u>. The search tactic must select the sensor device to be used (if the observer has more than one device available). It must choose between wide and narrow field of view and it must decide whether to scan across the field of search or to stare at specified points in the field of search. Some search tactics may involve switching between wide and narrow FOV or even switching from one sensor to another. In such a case the tactic must include decision logic to trigger the change. The tactic must also specify the acquisition level(s) which the observer is trying to attain.

5. <u>Compute Time to Detect</u>. Once the sensor device mode of use is specified, the NVL detection time model (or some other detection model) can be used to compute a time-to-detect for all or some subset of the targets in the potential target set. Times for switching devices or switching from wide to narrow FOV should also be included as appropriate. The search tactic must determine whether the acquisition times so computed for several targets are to be considered as having occurred simultaneously (as might be appropriate in a wide FOV search of a target-rich area) or sequentially (if a narrow FOV device is being used to stare at previously localized targets one at a time).

AD-A118 414 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 17/5
A TARGET ACQUISITION MODULE FOR THE STAR COMBINED ARMS COMBAT S--ETC(U)
APR 82 J K HARTMAN MIPR-CD-2-82
UNCLASSIFIED NPS55-82-014 NL

2 OF 2
AD A
118414

6. Schedule Detection Events. For targets whose acquisition time is small enough, a DETECT event must be scheduled by the search routines. The search tactics must specify the time threshold and perhaps a limit on how many targets can be acquired in one search cycle.

7. Schedule a New Search. Finally, the search tactic must decide when to terminate the current search event and thus the time at which the next SEARCH event for this observer should be scheduled to occur. Termination of the current search may occur because of an elapsed time threshold, or because of a limit on the number of targets acquired, or some combination of the two thresholds.

The variety of different computations which may be called for in a search event, and the options of multiple sensors and modes of employment make it unlikely that any single search tactic will be appropriate for all situations that we would like to simulate. Thus the approach to search tactics taken in STAR is to have several possible search tactics each represented by its own routine. Each tactic has parameters (such as the sensor device to be used) which customize it to a particular observer. New search tactics may be added by writing an appropriate new tactics routine without having to adjust the code for existing tactics.

B. CURRENT SEARCH TACTICS/NEW SEARCH TACTICS

The STAR Target Acquisition Module currently includes a number of search tactics routines designed to incorporate several detection models for various classes of searchers and targets. The following search tactics routines are available as of Dec 1981.

STK1 - Implements DYNTACS/ASARS visual detection model as used in original ground STAR Model, the original STAR Air Model, and the original Dismounted STAR Model.

STK2 - Single Sensor, single Mode of use NVL Detection Model.

STK3 - Air/Air Defense Visual Detection Model.

STK4 - Air Defense Radar Detection Model

Tactics STK1 and STK2 will be documented in this report with the emphasis being on STK2 as a multi-parameter search paradigm which can be customized to fit a wide variety of target acquisition situations. Documentation on STK3 and STK4 will be included with documetnation of the STAR Air/Air Defense Modules.

Other search tactics routines will be written as the need for other patterns of target acquisition behavior emerges. As each new tactic is added to the code, the changes required to use it are quite simple.

1. Add new STKn routine.

2. Add a call to the new STKn routine in event SEARCH.

3. Change the data set to include SEARCH TYPES which call for the new tactic and provide its parameters (if any).

4. Change the data set to cause combatants to use one of the newly defined SEARCH TYPES.

C. STK1 - DYNTACS VISUAL TARGET ACQUISITION.

The STK1 search tactics routine is included as a bridge to early versions of the STAR combat simulation which used the DYNTACS/ASARS visual target acquisition models. The situation modelled is unaided visual detection in a clear environment with daytime viewing conditions. It should be emphasized that this detection model does not interface with the STAR battlefield smoke model, and is thus inappropriate for any limited visibility

environment. Only one level of acquisition is modelled in the DYNTACS methodology and this is generally considered to correspond to "identification" in the NVL acquisition level.

The STK1 search tactic has no customizing parameters and is thus simple to use but of limited flexibility. The SIMSCRIPT programs for STK1 and for the VIS.DET.DYNTACS routine which it calls are included as Figure VI-1 and VI-2.

D. STK2 - NVL SINGLE SENSOR TARGET ACQUISITION

The STK2 search tactic is the first search tactics routine for STAR which was expressly written to approach our goals of modelling the interaction between a variety of sensor devices and sensor utilization patterns in limited visibility environments. The situation modelled is the use of a single NVL sensor device (including unaided visual search) over a short period of time called one search cycle (perhaps 30 seconds). At the end of a search cycle it is possible to switch to another device or another FOV mode for the next search cycle.

Search tactic STK2 has 16 parameters which can be used to customize the tactic routine to a particular individual combatant. These 16 parameters are defined for each SEARCH TYPE which uses tactic STK2 and are stored in the SCH.DATA array. Several different combatants (with different SEARCH TYPES) may simultaneously be using tactic STK2 with different parameters thus modelling different patterns of sensor device availability and/or utilization.

The 16 STK2 parameters are as follows:

1. TAC — Search tactic number (= 2 always for STK2)
2. SENSOR — SENSOR to use
3. MODE — Mode of use code for the sensor (wide vs narrow FOV)

```
ROUTINE STK1(A,OBS,TYPE) YIELDING TIME.USED, NEWTYPE
''****************************************************
''DYNTACS/ASARS VISUAL SEARCH MODEL
DEFINE PROC,C,X.LL.C,Y.LL.C,NEW.IND,IND,X.DIF,Y.DIF AS INTEGER VARIABLES
DEFINE A,B,OBS,TYPE,NEWTYPE AS INTEGER VARIABLES
DEFINE MAXDFRG,TIME.USED,SUPPTIME AS REAL VARIABLES
LET TIME.USED=DELTA.T
LET NEWTYPE = TYPE
''PRELIMINARY LIST MANAGEMENT
IF TIME.V GE 2.0 *DELTA.T AND DEFNUM(A) EQ 1
   CALL LIS.RELEASE(A)
   RETURN
OTHERWISE
LET LIST(*,*)=TRGT(1,NAME(A))
IF LIST(1,1)=0
   CALL CHG.SEC.SEARCH(A)
ALWAYS
LET LIST(*,*) = 0
''SCHEDULE DETECTIONS
IF PROLE(A) NE 0
   TRACE CALL SNAP.R  STOP  ''AIR/AD NOT USE THIS TACTIC
OTHERWISE      ''PROLE = 0 FOR GROUND ELEMENTS
LET MAXDFRG=GET(A,3,1)
IF GET.SP(A,4,0) NE 0
   LET SUPPTIME=TIM.SP(A)
ELSE
   LET SUPPTIME=0
ALWAYS
LET PROC=INT.F((MAXDFRG/BX.X.SZ)+.49)
CALL TERR.IND(A) YIELDING NEW.IND,X.LL.C,Y.LL.C
FOR X.DIF=MAX.F(0,X.LL.C-PROC) TO MIN.F(X.LL.C+PROC,BSE.N-1)  DO
FOR Y.DIF=MAX.F(0,Y.LL.C-PROC) TO MIN.F(Y.LL.C+PROC,HGHT.N-1) DO
  LET IND=Y.DIF*BSE.N+X.DIF +1
  FOR EACH C IN TERR.QUEUE(IND)  DO
    IF COLOR(C) EQ COLOR(A)  CYCLE  ELSE
    IF ALIVE.DEAD(C) NE 0  CYCLE  ELSE
      CALL VIS.DET.DYNTACS(A,C,MAXDFRG,SUPPTIME)
LOOP  LOOP  LOOP
GO TO OUT
'OUT'
RETURN
END
```

FIGURE VI-1   ROUTINE STK1

```
ROUTINE VIS.DET.DYNTACS(A,B,MAXDFRG,SUPPTIME)
''****************************************
DEFINE MAXDFRG,SUPPTIME AS REAL VARIABLES
DEFINE A,B,ACQ.LEV,ANSWER,POS AS INTEGER VARIABLES
DEFINE RX,RY AS REAL VARIABLES
DEFINE R,RN,PCT.VIS,DET.TIME,SIZE AS REAL VARIABLES
CALL LIS.CHECK(A,B) YIELDING ANSWER,ACQ.LEV,POS,SIZE
IF ANSWER EQ YES
   RETURN
OTHERWISE
LET RX=ABS.F(X.CURRENT(A)-X.CURRENT(B))
LET RY=ABS.F(Y.CURRENT(A)-Y.CURRENT(B))
IF RX GT MAXDFRG OR RY GT MAXDFRG
   RETURN
OTHERWISE
LET R=SQRT.F(RX*RX+RY*RY)
IF AREA(A) GT 359 LET ANSWER=YES
ELSE CALL SECTOR.CHECK(A,B) YIELDING ANSWER
ALWAYS
IF ANSWER EQ NO
   RETURN
OTHERWISE
LET RN=UNIFORM.F(0.0,1.0,7)
IF SYS.TYPE(B) NE 3
   CALL CARDIO(A,B,R,1.0,RN,MAXDFRG) YIELDING DET.TIME
ELSE
   CALL DISMTD.CARDIO(A,B,R,1.0,RN) YIELDING DET.TIME
ALWAYS
IF DET.TIME+SUPPTIME GT DELTA.T
   RETURN
OTHERWISE
LET FWD.LOOK=1
LET BWD.LOOK=0
CALL SIGHT (A,B)
LET PCT.VIS=MIN.F(PCB.VIS(1),PCB.UNC(1))*
   TARDIM(SYS.TYPE(B),WPN.TYPE(B),11)
IF PCT.VIS LT CRITICAL.VALUE
   RETURN
OTHERWISE
IF SYS.TYPE(B) NE 3
   CALL CARDIO(A,B,R,PCT.VIS,RN,MAXDFRG) YIELDING DET.TIME
ELSE
   CALL DISMTD.CARDIO(A,B,R,PCT.VIS,RN) YIELDING DET.TIME
ALWAYS
ADD SUPPTIME TO DET.TIME
IF DET.TIME LE DELTA.T
```

FIGURE VI-2   ROUTINE VIS.DET.DYNTACS

```
    SCHEDULE A DETECT(A,B,5,DET.TIME) IN DET.TIME UNITS
    IF COLOR(A) EQ DFNDR
       LET BSTD=DET.TIME
    ELSE
       LET RSTD=DET.TIME
    ALWAYS
 ALWAYS
 RETURN
 END
```

FIGURE VI-2   (CONTINUED)

4. LO.ACQ.LEV — Lowest acquisition level to consider

5. HI.ACQ.LEV — Highest acquisition level to try for

6. HFOS — Horizontal size of field of search (degrees)

7. VFOS — Vertical size of field of search (degrees)

8. MAXN — Maximum number of targets to acquire in one search cycle.

9. MAXTIME — Maximum time to spend in one search cycle

10. MINTIME — Minimum time to spend in one search cycle

11. SIMUL — Simultaneous (CODE = 1) vs Sequential (Code = 2) Acquisition times

12. FOVSW — FOV switch time to add for each target in sequential search

13. MAXEACH — Maximum time to spend on any single target

14. SOURCE — Source for Target: 1 = Battlefield, 2 = Own Detected List

15. PURGE — Purge Level for Detected List: 0 = NO PURGE

16. NEWTYPE — Search type to use for the next search cycle for this observer

A verbal description of the STK2 search tactic and its relation to these parameters is given in Volume I of this Report (Reference 1, Chapter IV-C). In this section we will go through the SIMSCRIPT code for Routine STK2 and several subroutines called by STK2.

1. Routine STK2

Purpose: Single sensor NVL search tactics routine

GIVEN ARGUMENTS

A	INTEGER	Pointer to entity doing the searching
OBS	INTEGER	Observer number on entity A
TYPE	INTEGER	SCH.TYPE to use for this search cycle

YIELDING ARGUMENTS

TIME.INC	REAL	Amount of time used by this search cycle
NEWTYPE	INTEGER	SCH.TYPE to be used for the next search cycle

LOCAL VARIABLES

SENSOR	INTEGER	Search Tactic Parameters as Defined Above
MODE	INTEGER	
LO.ACQ.LEV	INTEGER	
HI.ACQ.LEV	INTEGER	
HFOS	REAL	
VFOS	REAL	
MAXN	INTEGER	
MAXTIME	REAL	
SIMUL	INTEGER	
FOVSW	REAL	
MAXEACH	REAL	
SOURCE	INTEGER	
PURGE	INTEGER	
B	INTEGER	Pointer to potentially detectable target
N	INTEGER	Temporary variable
I	INTEGER	Loop index
MEM	INTEGER	Pointer to target memo entity
SUPPTIME	REAL	Suppression time for target acquisition
MXRNG	REAL	Maximum detection range for this system

GLOBAL VARIABLES

SCH.DATA	REAL 3-D	Search Tactics Parameters
LIST	INTEGER 2-D	A's Detection List
TRGT	INTEGER 2-D	Pointer to A's List
N.PO.TGT	INTEGER	Size of PO.TGT Set

ENTITY ATTRIBUTES FOR "TANK" ENTITIES

AREA	INTEGER	Horizontal search area
COLOR	INTEGER	ATKR or DFNDR
DEFNUM	INTEGER	DEFILADE Condition
NAME	INTEGER	ID Number
ALIVE.DEAD	INTEGER	0 if still alive, 1 if dead

ENTITY ATTRIBUTES FOR "TGT.MEMO" ENTITIES

PNTR	INTEGER	Pointer to the tank entity that is the potential target
RANKING	REAL	Minus target range - used as the ranking variable for the PO.TGT set.

SETS

BLUE.ALIVE	Alive DFNDR "Tank" entities
RED.ALIVE	Alive ATKR "Tank" entities
PO.TGT	TGT.MEMO entities for this search

ROUTINE AND FUNCTIONS CALLED

CHG.SEC.SEARCH	Change sector of search
DIM.F	Find array size
DIST	Compute distance from observer to target
EMPTY.PO.TGT	Empty PO.TGT set of all TGT.MEMO's
GET	Miscellaneous data filed by system/weapon type

INT.F	Integer
LIS.LEVEL.PURGE	Purge detected list of targets with low acquisition level
LIS.RELEASE	Totally erase detected list
NVL.1.PHASE	Do NVL Calculations
POT.TGT	Create TGT.MEMO's for PO.TGT Set
TIM.SP	Compute suppression time

EVENTS SCHEDULED

NONE	(NOTE: Detect events get scheduled in routine NVL.1.PHASE)

SIMSCRIPT CODE

See Figure VI-3

LINE BY LINE COMMENTARY

Lines 1 - 25	Declare the routine and define local variables.
Lines 26 - 32	Set yielding arguments and do a total suppresion check. If A is totally suppressed, then no detection computations will be attempted.
Lines 33 - 38	Erase the detected list for entities in full defilade and return.
Lines 39 - 43	Access A's detected list and change A's sector of search if the list is empty (indicating recent unsuccessful searching in the current search sector).
Lines 44 - 56	Compile a list of potentially detectable targets from a scan of the battlefield. TGT.MEMO entities for these targets are filed in the PO.TGT set.
Lines 57 - 70	Compile the PO.TGT set from A's own detected list, creating TGT.MEMO's for entities which are on A's list but at a lower acquisition level than desired.
Lines 72 - 76	Tally the size of the PO.TGT set for simulation summary statistics.
Lines 79 - 92	Set-up the parameters for the detection computations.

Lines 93 - 95 Perform the detection computations and schedule detect events by a call to NVL.1.PHASE.

Lines 96 - 102 Empty the PO.TGT set, purge the detected list of low level targets, and terminate the routine.

```
ROUTINE STK2(A,OBS,TYPE) YIELDING TIME.INC, NEWTYPE
''**************************************************
''SEARCH TACTIC TWO -- SINGLE PHASE NVL MODEL
''PARAMETERS FOR TACTIC 2 IN SCH.DATA(TYPE,OBS,J) ARE:
''      J=1    TACTIC NUMBER (=2)
''      J=2    NVL SENSOR CODE
''      J=3    MODE OF USE FOR SENSOR
''      J=4    LOWEST ACQUISITION LEVEL TO CONSIDER
''      J=5    HIGHEST ACQUISITION LEVEL TO CONSIDER
''      J=6    HORIZONTAL FIELD OF SEARCH
''      J=7    VERTICAL FIELD OF SEARCH
''      J=8    MAX # TARGETS TO ACQUIRE IN ONE PHASE
''      J=9    MAX TIME TO SPEND IN ONE PHASE
''      J=10   MIN TIME TO SPEND IN ONE PHASE
''      J=11   1=SIMULTANEOUS, 0=SEQUENTIAL ACQUISITION
''      J=12   FOV SWITCH TIME FOR EACH TGT IN SEQUENTIAL SEARCH
''      J=13   MAX TIME TO SPEND ON EACH TGT IN SEQUENTIAL SEARCH
''      J=14   SOURCE FOR TGTS, 1 = BATTLEFIELD, 2 = OWN LIST
''      J=15   PURGE LEVEL FOR DETECTED LIST, 0 = NO PURGE
''      J=16   SCH.TYPE TO USE FOR NEXT SEARCH EVENT
DEFINE PROC,C,X.LL.C,Y.LL.C,NEW.IND,IND,X.DIF,Y.DIF AS INTEGER VARIABLES
DEFINE A, OBS, TYPE, MAXN, SIMUL, B, SENSOR, MODE, N, I,
    MEM, NEWTYPE, PURGE, LO.ACQ.LEV, HI.ACQ.LEV AS INTEGER VARIABLES
DEFINE MAXTIME, MINTIME, TIME.INC, HPOS, VPOS AS REAL VARIABLES
DEFINE SUPPTIME, MXRNG, FOVSW, MAXEACH AS REAL VARIABLES
LET NEWTYPE = SCH.DATA(TYPE,OBS,16)
LET MAXTIME = SCH.DATA(TYPE,OBS,9)
'' TOTAL SUPPRESSION CHECK
LET SUPPTIME = TIM.SP(A)
IF MAXTIME LE SUPPTIME
    LET TIME.INC = MAXTIME
    RETURN
OTHERWISE
'' PRELIMINARY LIST MANAGEMENT
IF DEFNUM(A) EQ 1
    CALL LIS.RELEASE(A)
    LET TIME.INC = MAXTIME
    RETURN
OTHERWISE
LET LIST(*,*) = TRGT(1,NAME(A))
IF LIST(1,1) = 0
    CALL CHG.SEC.SEARCH(A)
ALWAYS
''CREATE POTENTIAL TARGET SET
LET MXRNG = GET(A,3,1)
IF INT.F(SCH.DATA(TYPE,OBS,14)) EQ 1    ''TGTS FROM BATTLEFIELD
```

FIGURE VI-3   ROUTINE STK2

```
LET PROC=INT.F((MXRNG/BX.X.SZ)+.49)
CALL TERR.IND(A) YIELDING NEW.IND,X.LL.C,Y.LL.C
FOR X.DIF=MAX.F(0,X.LL.C-PROC) TO MIN.F(X.LL.C+PROC,BSE.N-1)  DO
FOR Y.DIF=MAX.F(0,Y.LL.C-PROC) TO MIN.F(Y.LL.C+PROC,HGHT.N-1)  DO
  LET IND=Y.DIF*BSE.N+X.DIF+1
  FOR EACH C IN TERR.QUEUE(IND)  DO
    IF COLOR(C) EQ COLOR(A)  CYCLE  ELSE
    IF ALIVE.DEAD(C) NE 0  CYCLE  ELSE
      CALL POT.TGT(A,C,MXRNG)
LOOP  LOOP  LOOP
ELSE              ''TGTS FROM OWN LIST
    LET HI.ACQ.LEV = SCH.DATA(TYPE,OBS,5)
    IF LIST(1,1) NE 0
        LET N = DIM.F(LIST(1,*))
        FOR I = 1 TO N DO
            IF LIST(2,I) LT HI.ACQ.LEV
                CREATE A TGT.MEMO CALLED MEM
                LET PNTR(MEM) = LIST(1,I)
                LET RANKING(MEM) = -1.0 * DIST(A,LIST(1,I))
                FILE MEM IN PO.TGT
            ALWAYS
        LOOP
    ALWAYS
ALWAYS
LET LIST(*,*) = 0
LET N = N.PO.TGT
IF N GT 0
    LET NZPOT = N    ''FOR TALLY
ALWAYS
LET NUMPOT = N       ''FOR TALLY
''DETECTION COMPUTATIONS
''SETUP SEARCH PARAMETERS
LET SENSOR = SCH.DATA(TYPE,OBS,2)
LET MODE = SCH.DATA(TYPE,OBS,3)
LET LO.ACQ.LEV = SCH.DATA(TYPE,OBS,4)
LET HI.ACQ.LEV = SCH.DATA(TYPE,OBS,5)
LET HFOS = SCH.DATA(TYPE,OBS,6)
IF HFOS LT 0.01
    LET HFOS = AREA(A)
ALWAYS
LET VFOS = SCH.DATA(TYPE,OBS,7)
LET MAXN = SCH.DATA(TYPE,OBS,8)
LET MINTIME = SCH.DATA(TYPE,OBS,10)
LET SIMUL = SCH.DATA(TYPE,OBS,11)
LET FOVSW = SCH.DATA(TYPE,OBS,12)
LET MAXEACH = SCH.DATA(TYPE,OBS,13)
```

FIGURE VI-3 (CONTINUED)

```
CALL NVL.1.PHASE(A,SENSOR,MODE,LO.ACQ.LEV,HI.ACQ.LEV,HPOS,VPOS,MAXN,
    MAXTIME,MINTIME,SIMUL,SUPPTIME,POVSW,MAXEACH)
    YIELDING TIME.INC
''CLEANUP
CALL EMPTY.PO.TGT
LET PURGE = SCH.DATA(TYPE,OBS,15)
IF PURGE GT 0
    CALL LIS.LEVEL.PURGE(A,PURGE)
ALWAYS
RETURN
END
```

FIGURE VI-3 (CONTINUED)

2. ROUTINE NVL.1.PHASE

Purpose: This routine is responsible for keeping track of the time during a search event. In particular it distinguishes between the simultaneous and sequential target acquisiton modes, schedules DETECT events when appropriate, and computes the total TIME.USED by a search cycle.

GIVEN ARGUMENTS

A	INTEGER	Pointer to entity doing the searching
SENSOR	INTEGER	
MODE	INTEGER	
LO.ACQ.LEV	INTEGER	
HI.ACQ.LEV	INTEGER	
HFOS	REAL	
VFOS	REAL	Tactic parameters as defined above
MAXN	INTEGER	
MAXTIME	REAL	
MINTIME	REAL	
SIMUL	INTEGER	
FOVSW	REAL	
MAXEACH	REAL	
TIMSP	REAL	Suppresion time for target acquisition

YIELDING ARGUMENT

TIME.USED	REAL	Amount of TIME.USED by this search cycle

LOCAL VARIABLE

B	INTEGER	Pointer to potential target entities
N	INTEGER	Size of PO.TGT set

LOCAL VARIABLES CONTINUED

I	INTEGER	Loop index ranging from 1 to N
MEMO	INTEGER	Pointer to target MEMO entities from PO.TGT set
NACQ	INTEGER	Number of targets acquired so far in this cycle
ACQ.LEV	INTEGER	Acquisition level achieved for a target
ANS	INTEGER	Yes/No, is B already on A's list?
OLD.LEV	INTEGER	If B is on A's list, the acquisition level
POS	INTEGER	If B is on A's list, the position in the list
ACQ.TIM	REAL	Time required to acquire target at level ACQ.LEV
SIZE	REAL	Return from LIS.CHECK - not used.

GLOBAL VARIABLES

BSTD	REAL	Accumulation variables for detection time statistics
RSTD	REAL	
DFNDR	INTEGER = 1	
YES	INTEGER = 1	
NO	INTEGER = 0	
N.PO.TGT	INTEGER	Size of PO.TGT Set

ENTITY ATTRIBUTE FOR "TANK" ENTITIES

COLOR	INTEGER	ATKR/DFNDR

ENTITY ATTRIBUTE FOR "TGT.MEMO" ENTITIES

PNTR	INTEGER	Pointer to target entity B

SET

PO.TGT	Set of Target MEMOS from routine STK2

ROUTINES AND FUNCTIONS CALLED

LIS.CHECK	To see if B is already on A's list
MAX.F	Maximum
MIN.F	Minimum
NVL.DET	To compute acquisition time and level for each single target

EVENT SCHEDULED

DETECT	Detection event

SIMSCRIPT CODE

See Figure VI-4

LINE BY LINE COMMENTARY

Lines 1 - 23	Declare the routine and define variables.
Lines 24 - 25	Initialize time and acquisitions to zero.
Lines 26 - 29	Take care of the case where there are no potential targets.
Lines 30 - 33	Start the main loop over all potential targets in order of their ranking attributes. this loop continues as long as the number of detectevents scheduled is less than MAXN.
Lines 34 - 35	Screen out targets that have already been acquired at the desired level.
Lines 36 - 38	Call NVL.DET to compute the acquisiiton time and acquisition level for the current target.
Lines 39 - 53	Coordinate timing for simultaneous searching. If ACQ.TIM is less than MAXTIME then a detection event is scheduled. TIME.USED is set to the largest ACQ.TIM encountered.
Lines 54 - 75	Coordinate timing for sequential searching. The ACQ.TIM's are accumulated to give TIME.USED. If ACQ.TIM is less than MAXEACH and TIME.USED is less than MAXTIME, then a detection event is scheduled.
Line 77	Destroys the TGT.MEMO entity for this potential target.
Lines 79 - 80	Make sure that TIME.USED is between MINTIME and MAXTIME.

```
ROUTINE NVL.1.PHASE
''**********************************
GIVEN     A,              ''OBSERVER POINTER
          SENSOR,         ''SENSOR DEVICE CODE
          MODE,           ''MODE OF SENSOR USE
          LO.ACQ.LEV,     ''LOWEST ACQUISITION LEVEL TO CONSIDER
          HI.ACQ.LEV,     ''HIGHEST ACQUISITION LEVEL TO CONSIDER
          HFOS,           ''HORIZONTAL FIELD OF SEARCH
          VFOS,           ''VERTICAL FIELD OF SEARCH
          MAXN,           ''MAX # TGTS TO ACQUIRE IN ONE PHASE
          MAXTIME,        ''MAX TIME TO SPEND IN THIS PHASE
          MINTIME,        ''MIN TIME TO SPEND IN THIS PHASE
          SIMUL,          ''YES=SIMULTANEOUS, NO=SEQUENTIAL
          TIMSP,          ''SUPPRESSION TIME TO BE ADDED TO EACH DETECT TIME
          FOVSW,          ''SWITCH TIME FOR EACH TGT IN SEQUENTIAL SEARCH
          MAXEACH         ''MAX TIME FOR EACH TGT IN SEQUENTIAL SEARCH
YIELDING TIME.USED  ''BY THIS SEARCH PHASE
''PROCESSES TARGETS FROM PO.TGT SET.  EACH SUCCESSFUL ACQUISITION
''SCHEDULES A DETECTION EVENT
DEFINE A, SENSOR, MODE, MAXN, SIMUL, B, N, I, MEMO, NACQ, ACQ.LEV,
    ANS, OLD.LEV, POS, LO.ACQ.LEV, HI.ACQ.LEV AS INTEGER VARIABLES
DEFINE ACQ.TIM, MAXTIME, MINTIME, HFOS, VFOS, TIME.USED, SIZE, TIMSP,
    FOVSW, MAXEACH AS REAL VARIABLES
LET TIME.USED = 0.0
LET NACQ = 0
LET N = N.PO.TGT
IF N = 0
    LET TIME.USED = MAXTIME
    RETURN
OTHERWISE
FOR I = 1 TO N WHILE NACQ LT MAXN DO
    REMOVE THE FIRST MEMO FROM PO.TGT
    LET B = PNTR(MEMO)
    CALL LIS.CHECK(A,B) YIELDING ANS,OLD.LEV,POS,SIZE
    IF ANS EQ NO OR OLD.LEV LT HI.ACQ.LEV
        CALL NVL.DET(A,B,SENSOR,MODE,LO.ACQ.LEV,HI.ACQ.LEV,HFOS,VFOS,
            MAXEACH)  YIELDING ACQ.TIM,ACQ.LEV
        ADD TIMSP TO ACQ.TIM
        IF SIMUL EQ YES     '' SIMULTANEOUS SEARCH FOR SEVERAL TARGETS
            IF ACQ.TIM LE MAXTIME
                LET TIME.USED = MAX.F(TIME.USED,ACQ.TIM)
                IF ANS EQ NO OR OLD.LEV LT ACQ.LEV
                    SCHEDULE A DETECT(A,B,ACQ.LEV,ACQ.TIM) IN ACQ.TIM UNITS
                    ADD 1 TO NACQ
                    IF COLOR(A) EQ DFNDR
                        LET BSTD = ACQ.TIM
```

FIGURE VI-4   ROUTINE NVL.1.PHASE

```
                    ELSE
                        LET RSTD = ACQ.TIM
                    ALWAYS
                ALWAYS
            ELSE
                LET TIME.USED = MAXTIME
            ALWAYS
        ELSE           ''SEQUENTIAL SEARCH FOR SEVERAL TARGETS
            ADD FOVSW TO TIME.USED
            IF ACQ.TIM GT MAXEACH
                ADD MAXEACH TO TIME.USED
                DESTROY THE TGT.MEMO CALLED MEMO
                CYCLE
            OTHERWISE
            ADD ACQ.TIM TO TIME.USED
            IF TIME.USED GT MAXTIME
                DESTROY THE TGT.MEMO CALLED MEMO
                LEAVE  ''THE LOOP SINCE TIME IS UP
            OTHERWISE
            IF ANS EQ NO OR OLD.LEV LT ACQ.LEV
                SCHEDULE A DETECT(A,B,ACQ.LEV,TIME.USED) IN TIME.USED UNITS
                ADD 1 TO NACQ
                IF COLOR(A) EQ DFNDR
                    LET BSTD = ACQ.TIM
                ELSE
                    LET RSTD = ACQ.TIM
                ALWAYS
            ALWAYS
        ALWAYS
    ALWAYS
    DESTROY THE TGT.MEMO CALLED MEMO
LOOP
LET TIME.USED = MIN.F(TIME.USED,MAXTIME)
LET TIME.USED = MAX.F(TIME.USED,MINTIME)
RETURN
END
```

FIGURE VI-4   (CONTINUED)

3. ROUTINE POT.TGT

Purpose: Routine POT.TGT does range and sector checks to screen potentially detectable elements. TGT.MEMO entities are created for elements which pass the screen and are filed in the PO.TGT set in order of closest range.

GIVEN ARGUMENTS

A	INTEGER	Observer entity
B	INTEGER	Potential target entity
MXRNG	REAL	Detection range limit

LOCAL VARIABLES

ANS	INTEGER	Return answer from SECTOR.CHECK
MEM	INTEGER	Pointer to TGT.MEMO entity
RX	REAL	Range in X - Coordinate
RY	REAL	Range in Y - Coordinate
RNG	REAL	Range from A to B

GLOBAL VARIABLES

YES	INTEGER	= 1

ENTITY ATTRIBUTES FOR "TANK" ENTITIES

DEFNUM	INTEGER	Defilade condition of target
X.CURRENT	REAL	X battlefield coordinates
Y.CURRENT	REAL	Y battlefield coordinates

ENTITY ATTRIBUTES FOR "TGT.MEMO" ENTITIES

PNTR	INTEGER	Pointer to potential target B
RANKING	REAL	Minus range between A and B

SET

PO.TGT	Set of TGT.MEMO"s for potential targets ranked on high RANKING.

ROUTINE AND FUNCTIONS

SQRT.F	Square root
SECTOR.CHECK	Checks whether B is in A's search sector

EVENTS SCHEDULED

None

SIMSCRIPT CODE

See Figure VI-5

COMMENTARY

Self-Explanatory

```
ROUTINE POT.TGT(A,B,MXRNG)
''*********************************************************************
''CHECKS WHETHER B MIGHT POTENTIALLY BE A CANDIDATE FOR ACQUISITION BY A.
''   IF SO, FILES A TARGET MEMO FOR B IN THE PO.TGT SET
DEFINE A, B, ANS, MEM AS INTEGER VARIABLES
DEFINE RX, RY, RNG, MXRNG AS REAL VARIABLES
IF DEFNUM(B) NE 1
    LET RX = ABS.F(X.CURRENT(A)-X.CURRENT(B))
    LET RY = ABS.F(Y.CURRENT(A)-Y.CURRENT(B))
    IF RX LT MXRNG AND RY LT MXRNG
        LET RNG = SQRT.F(RX*RX + RY*RY)
        IF RNG LT MXRNG      ''MAX ACQUISITION RANGE INDEPT OF SENSOR
            CALL SECTOR.CHECK(A,B) YIELDING ANS
            IF ANS EQ YES
                CREATE A TGT.MEMO CALLED MEM
                LET PNTR(MEM) = B
                LET RANKING(MEM) = -RNG
                FILE MEM IN PO.TGT
            ALWAYS
        ALWAYS
    ALWAYS
ALWAYS
RETURN
END
```

FIGURE VI-5   ROUTINE POT.TGT

4. ROUTINE EMPTY.PO.TGT

Purpose: Empty the potential target set at the end of a search cycle by one observer so that it can be used by the next observer to search. The SIMSCRIPT code is given in Figure VI-6. No further explanation should be required.

```
ROUTINE EMPTY.PO.TGT
''**********************
DEFINE MEMO, I, N AS INTEGER VARIABLES
LET N = N.PO.TGT
FOR I = 1 TO N DO
   REMOVE THE FIRST MEMO FROM PO.TGT
   DESTROY THE TGT.MEMO CALLED MEMO
LOOP
RETURN N
END
```

FIGURE VI-6 ROUTINE EMPTY.PO.TGT

5. USE OF THE STK2 SEARCH TACTIC

The reader is referred to Volume I of this report (Reference 1, Chapter IV, Section D) for an example of applying STK2. The situation modelled is that of an observer using unaided visual search to make a survey of his search sector looking for anything that might be a military target. He then uses field glasses to focus on each detected target in succession in an attempt to identify the target.

VII. CONCLUSIONS

This report presents detailed documentation for the STAR Target Acquisition Module. The Target Acquisition Module has been developed to enable users of STAR to simulate a variety of sensor devices and sensor utilization patterns in limited visibility conditions. The module is designed to be easily enhanced as the need for additional search tactics becomes apparent. For further discussion of the Target Acquisition Module see VOLUME I of this report (Reference 1) and also the STAR Smoke Model documentation (Reference 2).

REFERENCES

1. Hartman, J. K. "A Target Acquisition Module for the STAR Combined Arms Combat Simulation Model, Volume I, Naval Postgraduate School, Technical Report NPS-55-82-001, January 1982.

2. Carpenter, H. J. and Hartman, J. K. "The STAR Battlefield Smoke Module", (Forthcoming)

3. Hartman, J. K. "Parametric Terrain and Line of Sight Modelling in the STAR Combat Model", Naval Postgraduate School, Technical Report NPS 55-79-018, August 1979.

DISTRIBUTION LIST

	NO. OF COPIES
Library, Code 0142 Naval Postgraduate School Monterey, CA 93940	4
Dean of Research Code 012 Naval Postgraduate School Monterey, CA 93940	1
Library, Code 55 Naval Postgraduate School Monterey, CA 93940	1
Professor James K. Hartman Code 55Hh Naval Postgraduate School Monterey, CA 93940	100
Defense Technical Information Center ATTN: DTICDDR Cameron Station Alexanria, Virginia 22314	2